What’s New in Instruments

Session 411

These are confidential sessions—please refrain from streaming, blogging, or taking pictures.

What’s New in Instruments

Lynne Salameh
Performance Tools Engineer

Today's Agenda

Intro

Time Profiler

Multicore

Zombies

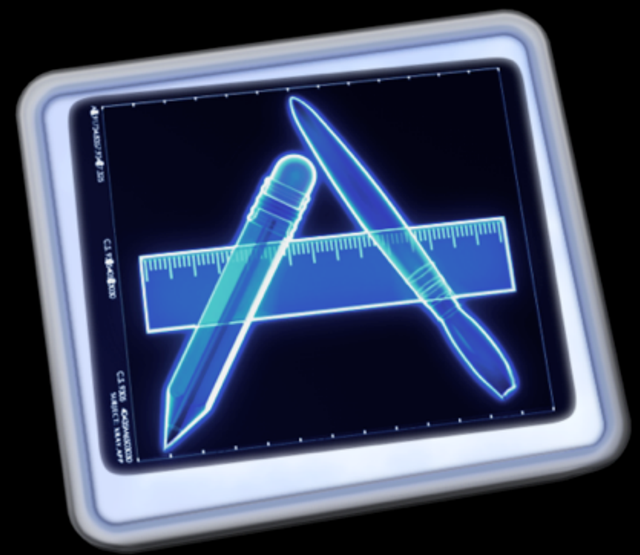
Object Graph

Other Features

Introduction

Instruments in Snow Leopard

- Enhanced data collection, mining, and display
- New instrumentation
- Improved performance



Intro
Time Profiler
Multicore
Zombies
Object Graph
Other Features

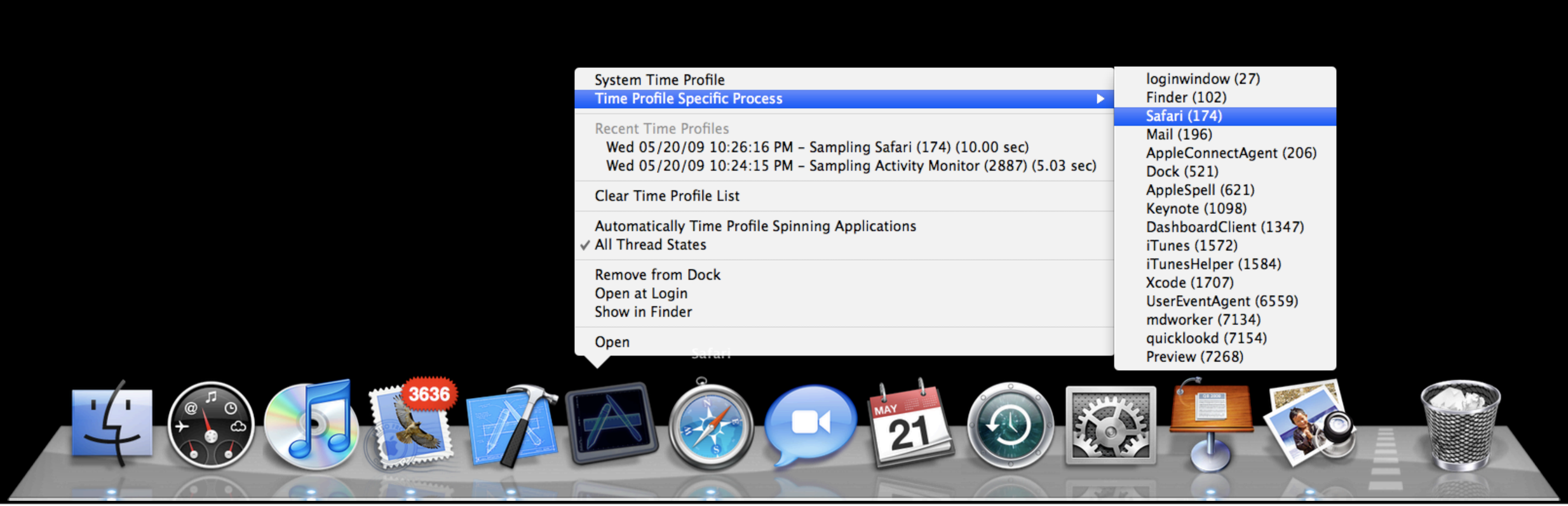
Demo

Time Profiler

Time Profiler

What is it?

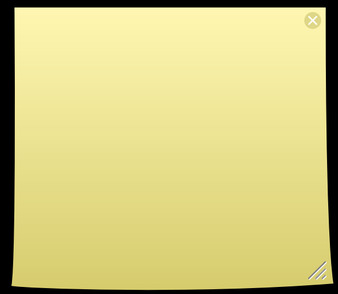
- Next generation statistical time sampler
- In Instruments or standalone
- Profiles one or all processes



Time Profiler

Launching

- Dock
  - Automatic spin profiling
  - Global hot keys
  - Special file format
    - (Compact, not symbolicated)
- Within Instruments



Time Profiler

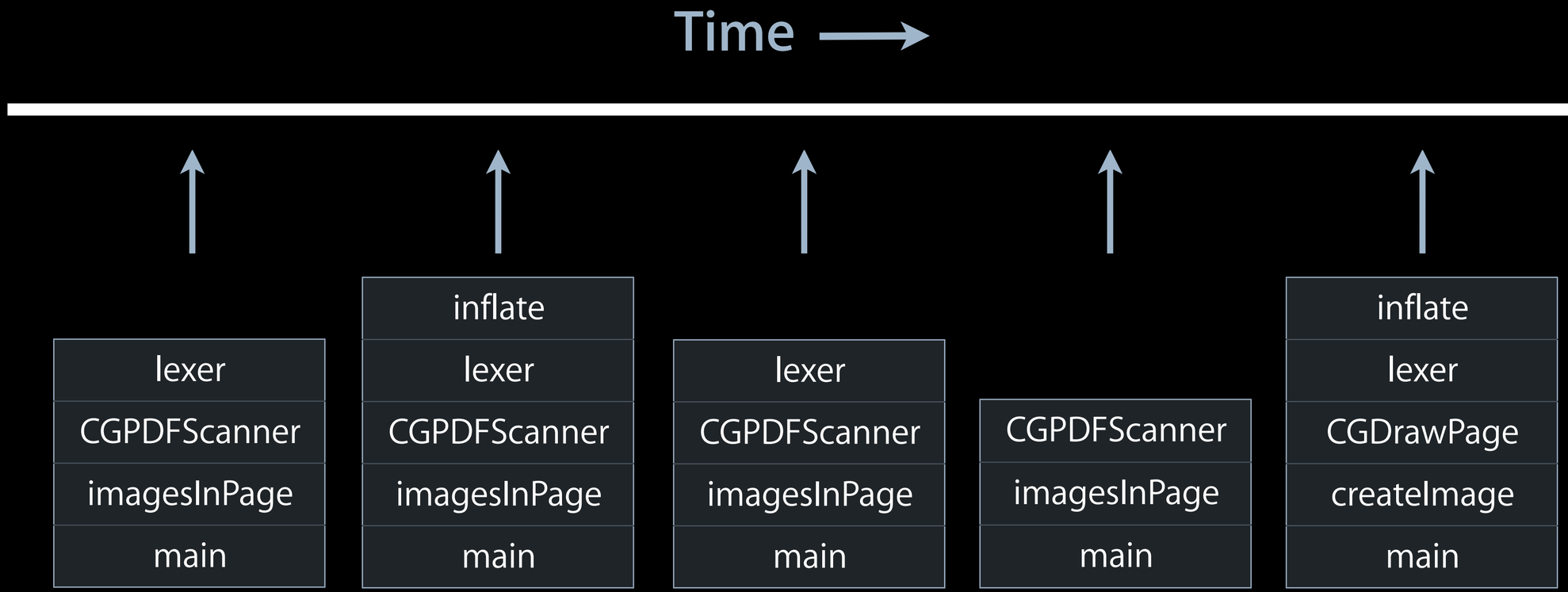
Advantages

- Low impact
- Performed within kernel
- All vs. running thread states
- System profiling

Time Profiler

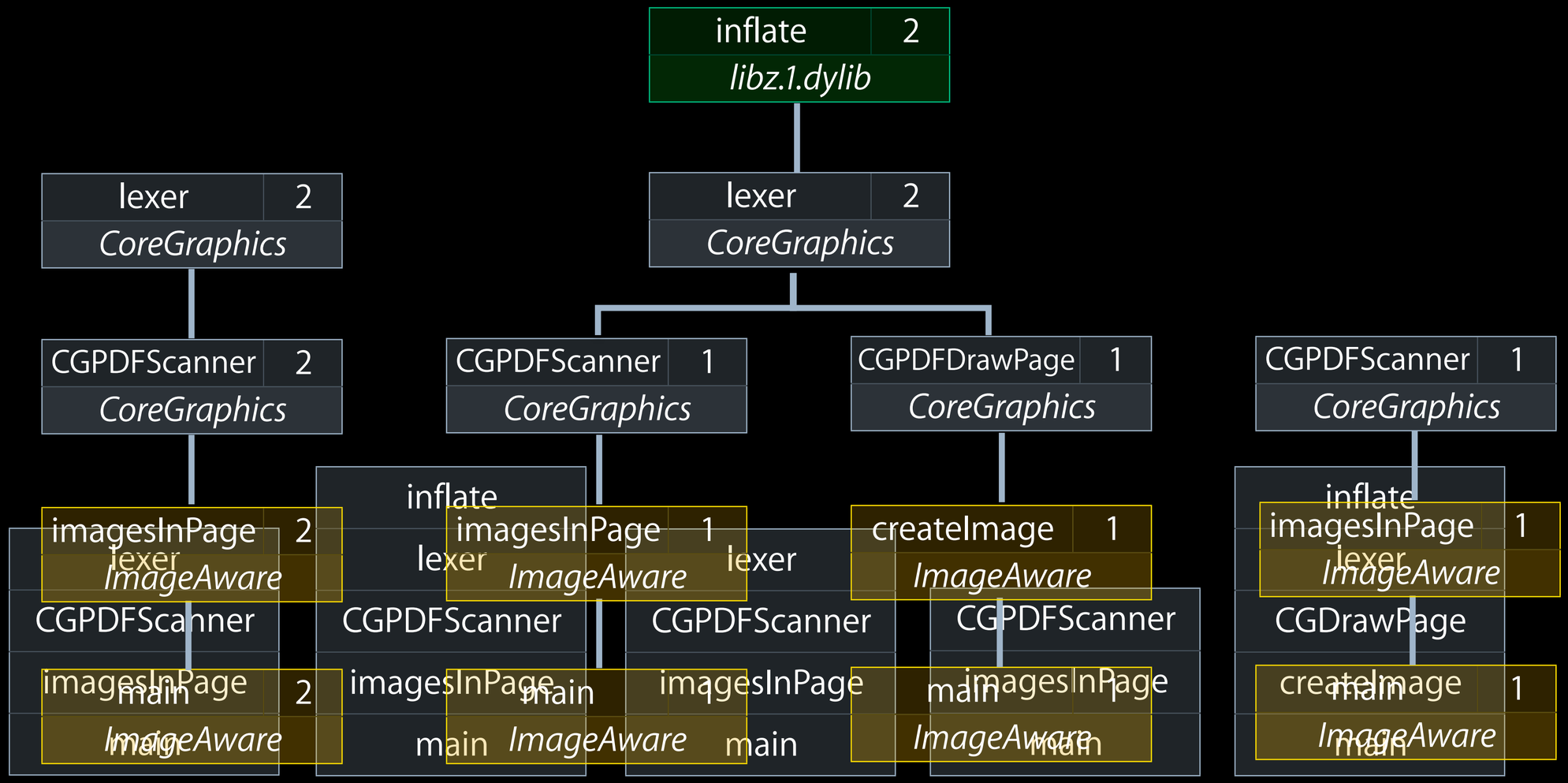
How does it work?

- Periodic samples of threads
- Gathers call stacks
  - Including kernel time and drivers



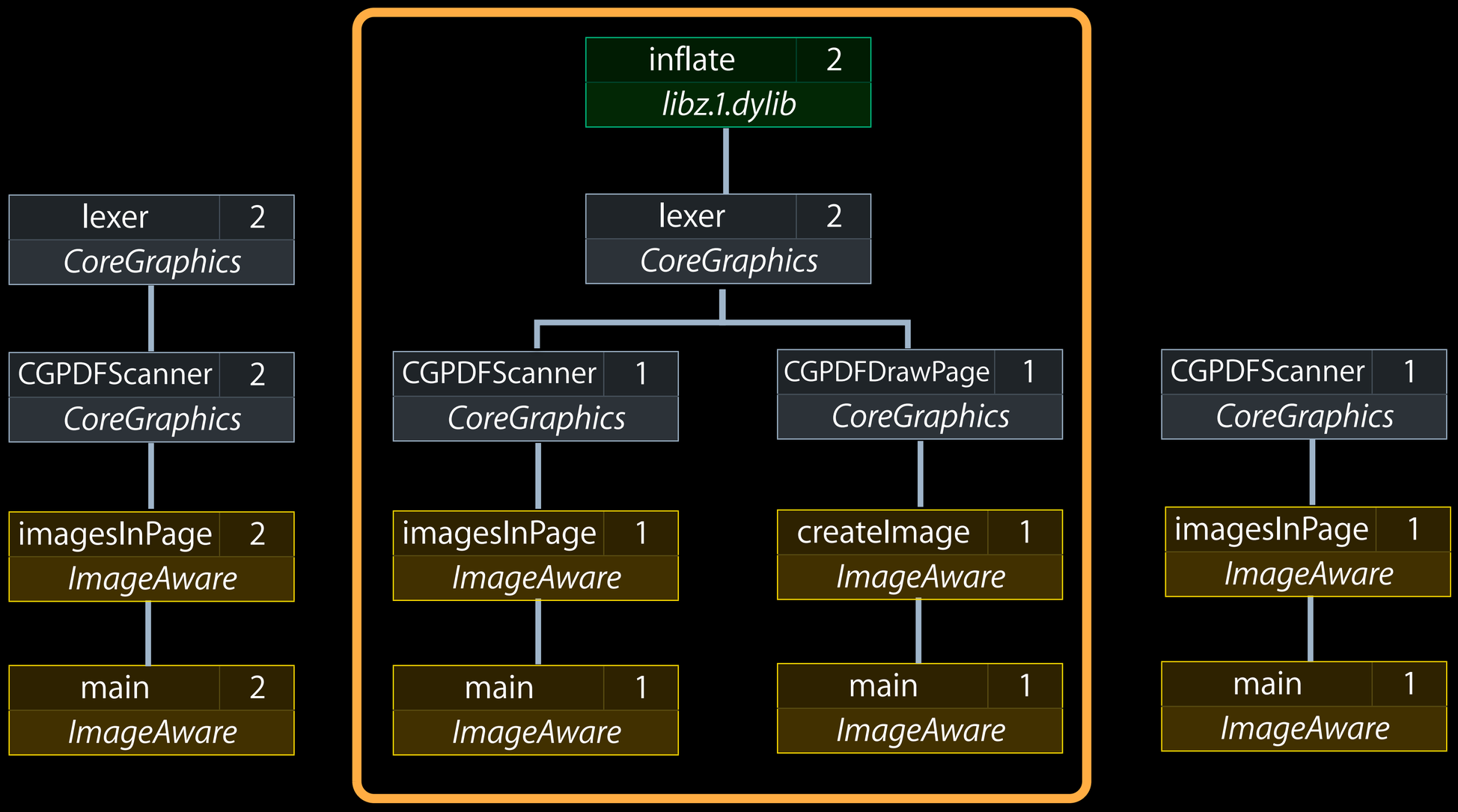
Time Profiler

Building Call Tree



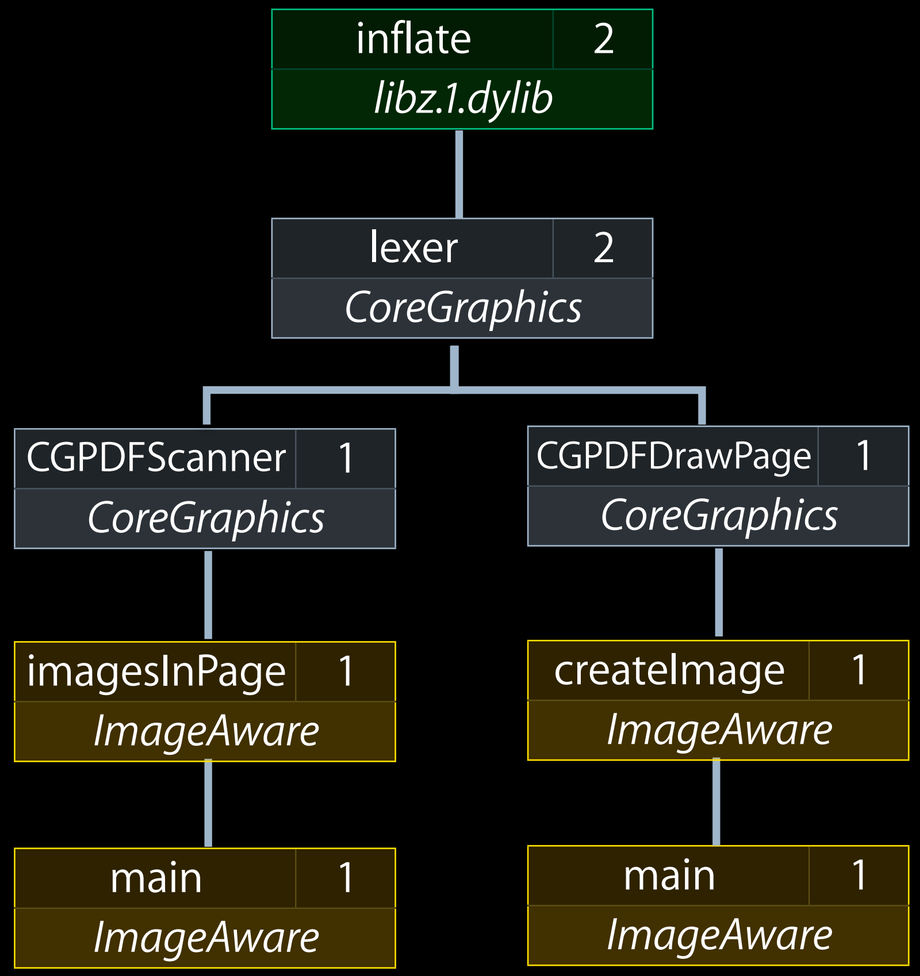
Time Profiler

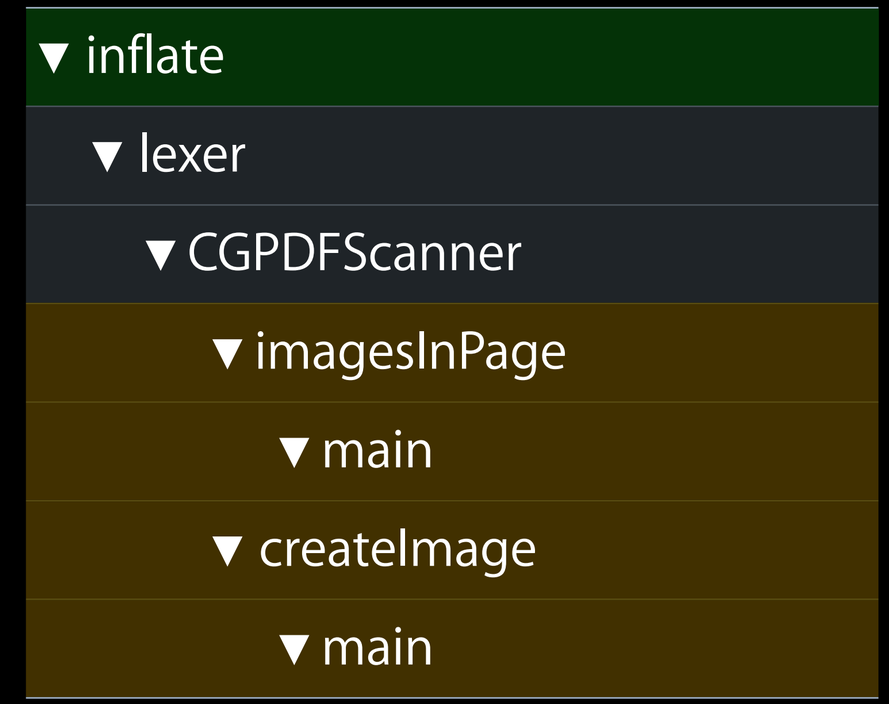
Building Call Tree



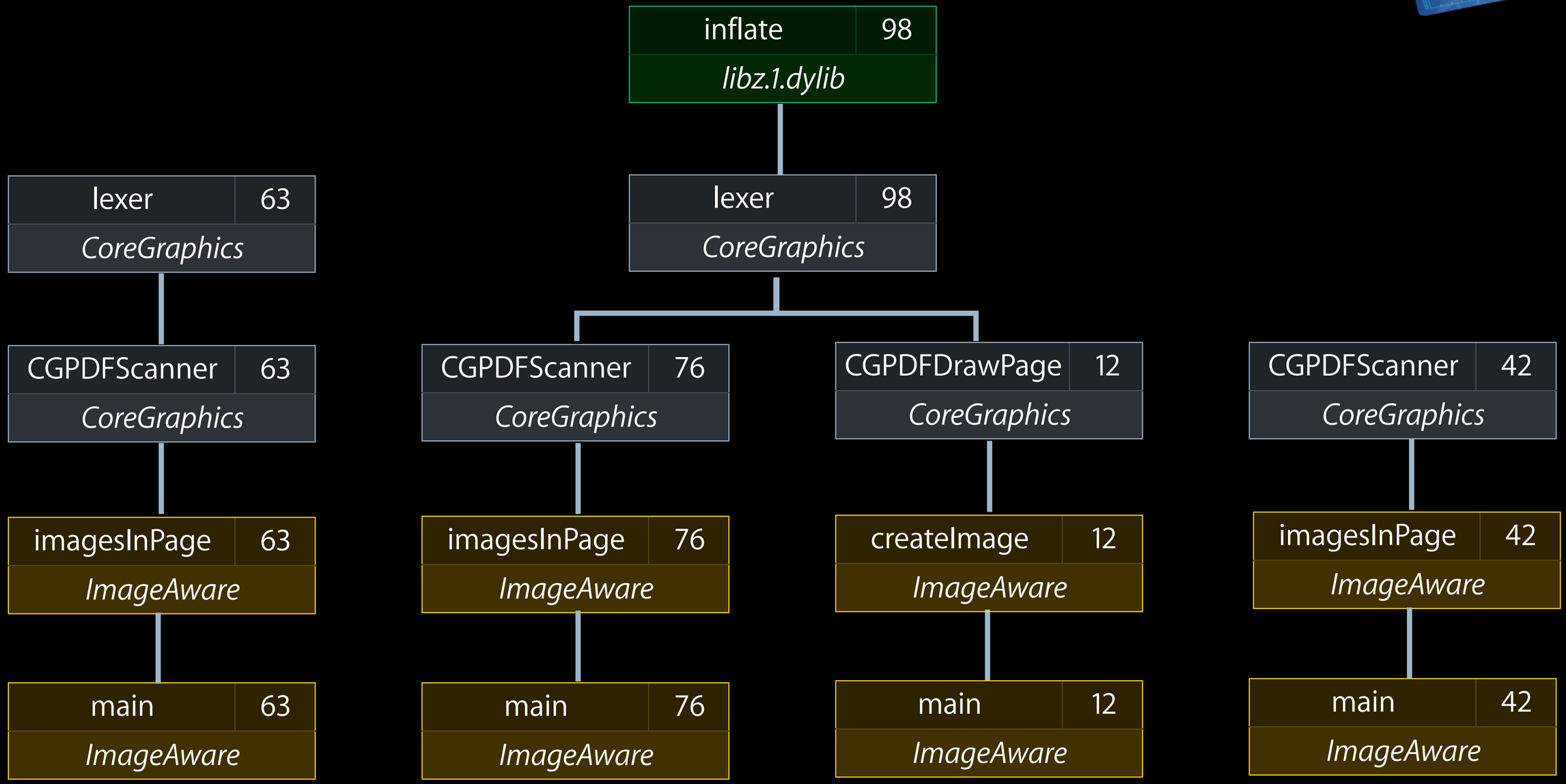
Time Profiler

Building Call Tree

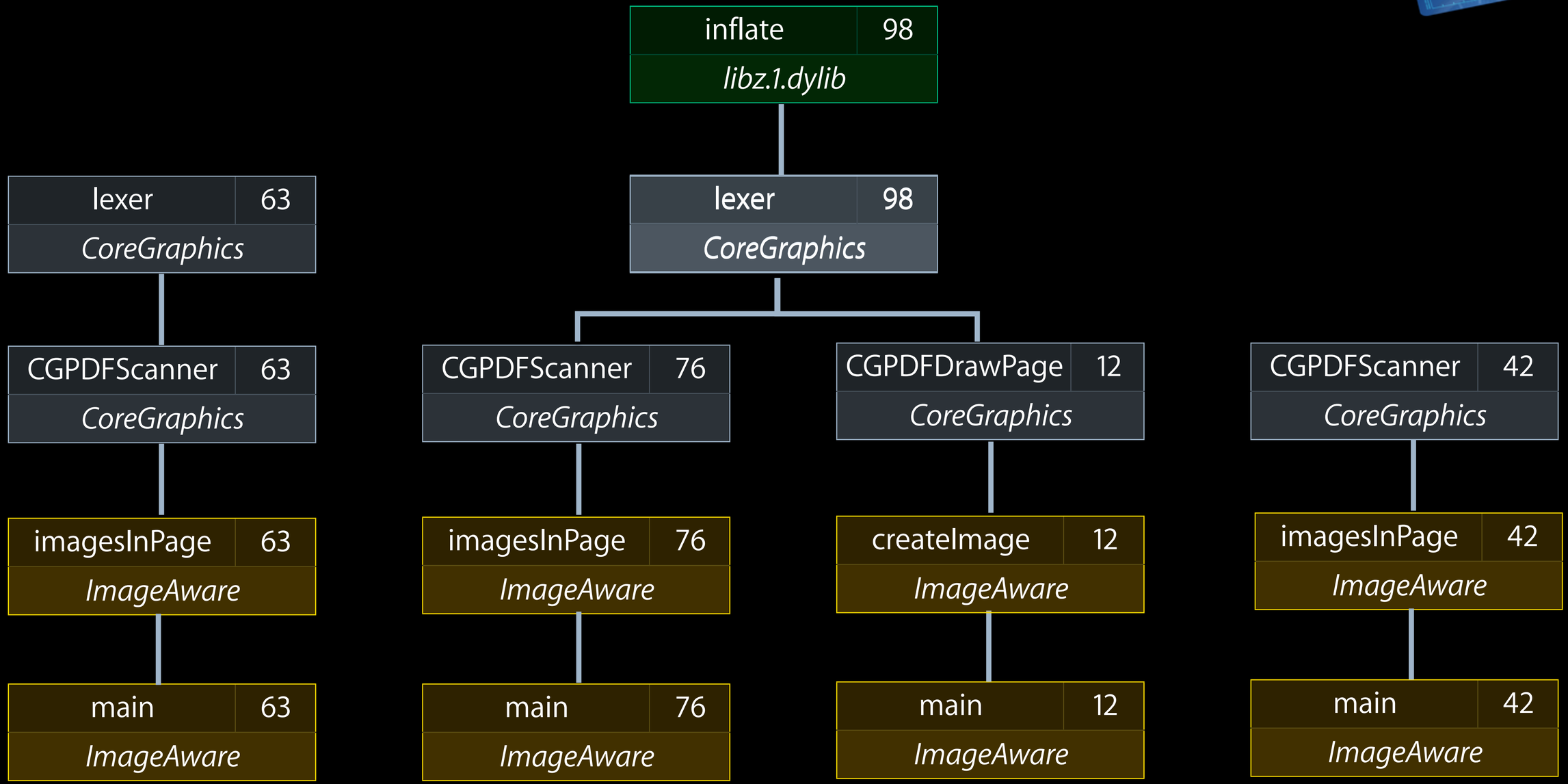




Data Mining

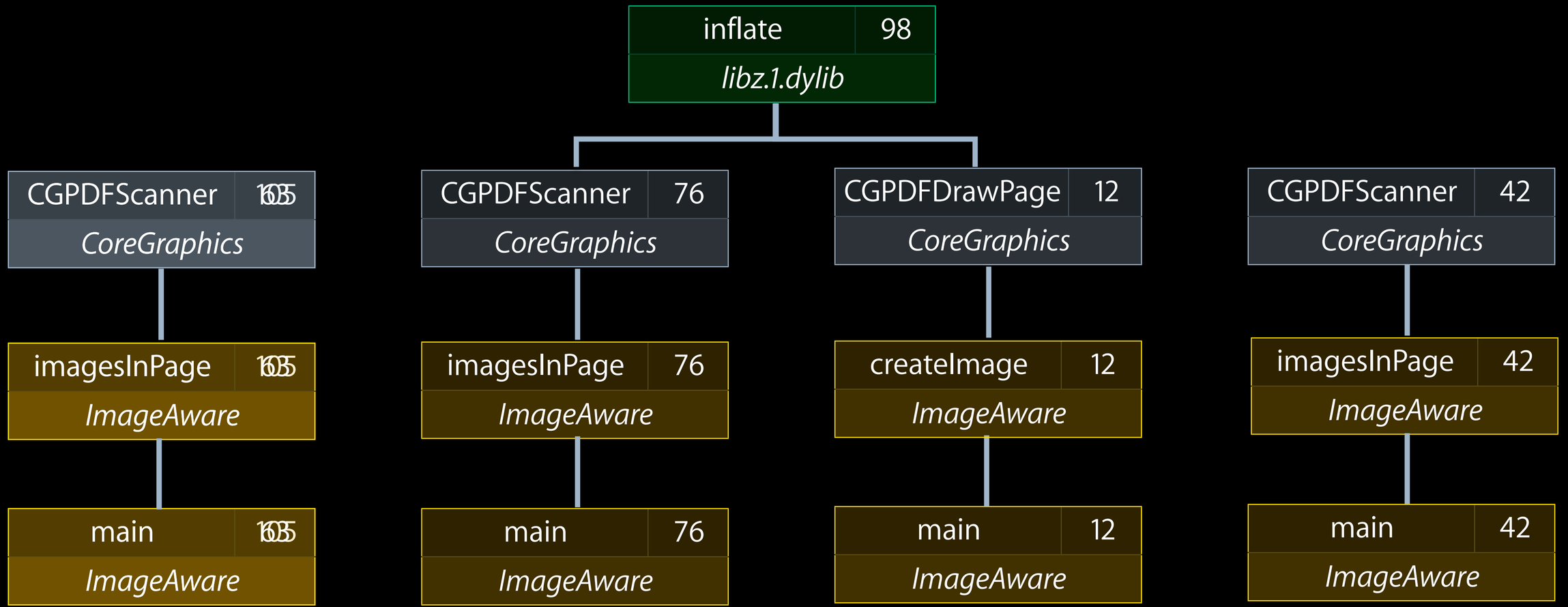

inflate 98
libz.1.dylib
lexer 63
CoreGraphics
lexer 98
CoreGraphics
CGPDFScanner 63
CoreGraphics
CGPDFScanner 76
CoreGraphics
CGPDFDrawPage 12
CoreGraphics
CGPDFScanner 42
CoreGraphics
imagesInPage 63
ImageAware
imagesInPage 76
ImageAware
createImage 12
ImageAware
imagesInPage 42
ImageAware
main 63
ImageAware
main 76
ImageAware
main 12
ImageAware
main 42
ImageAware

Data Mining



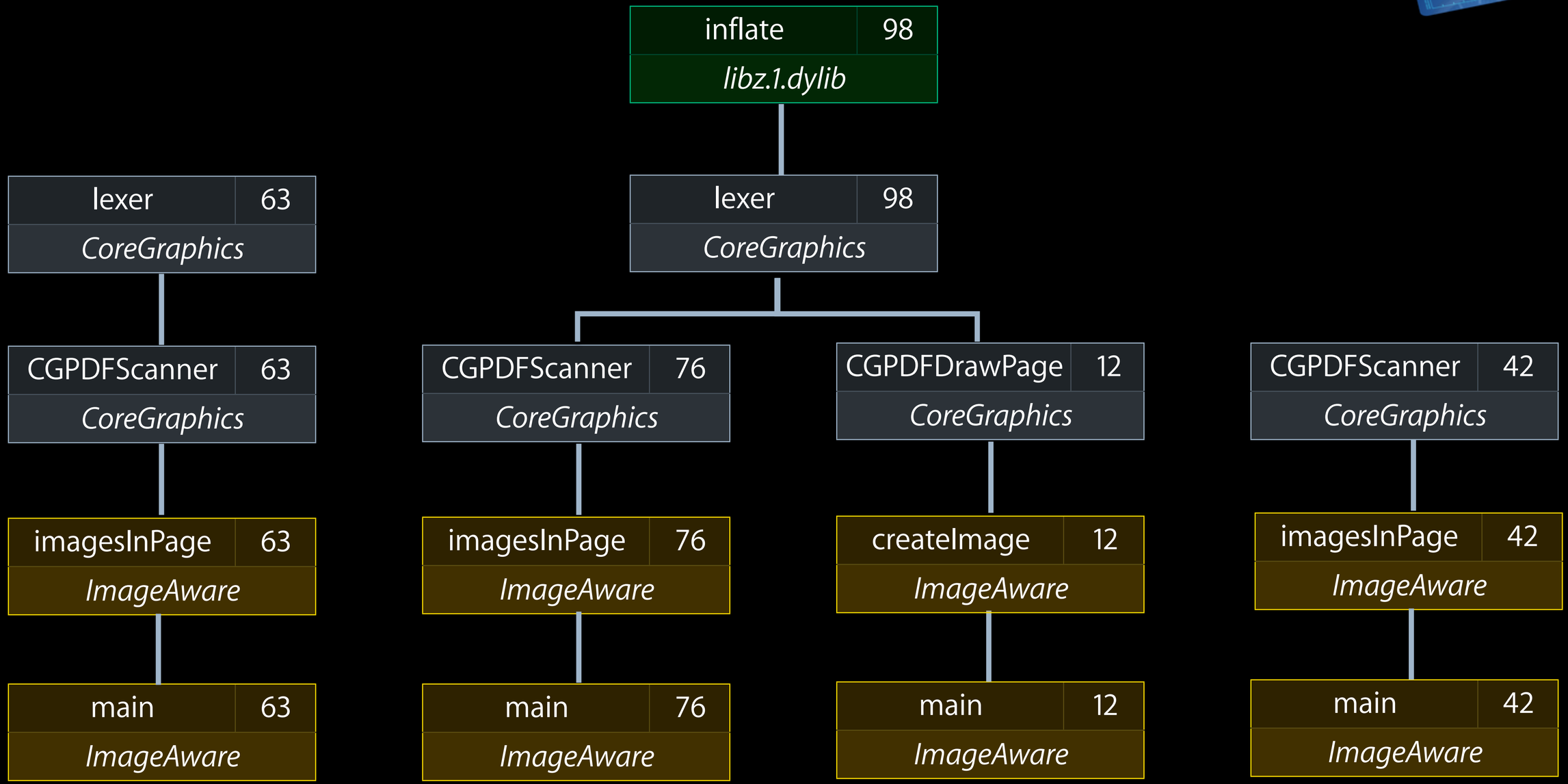
Charge Symbol to Caller

Data Mining



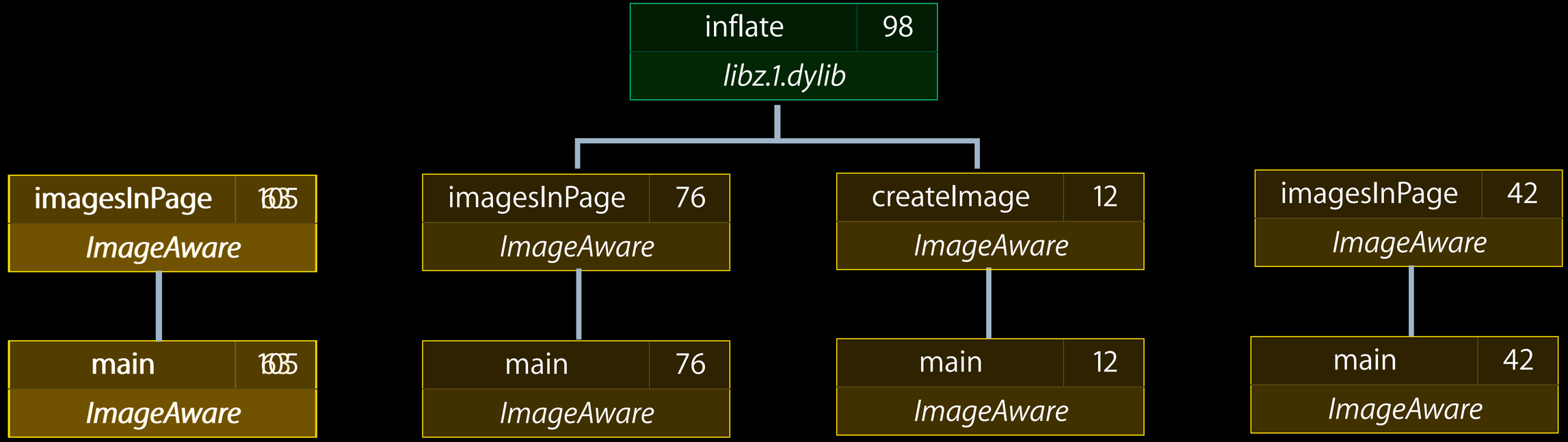
Charge Symbol to Caller

Data Mining



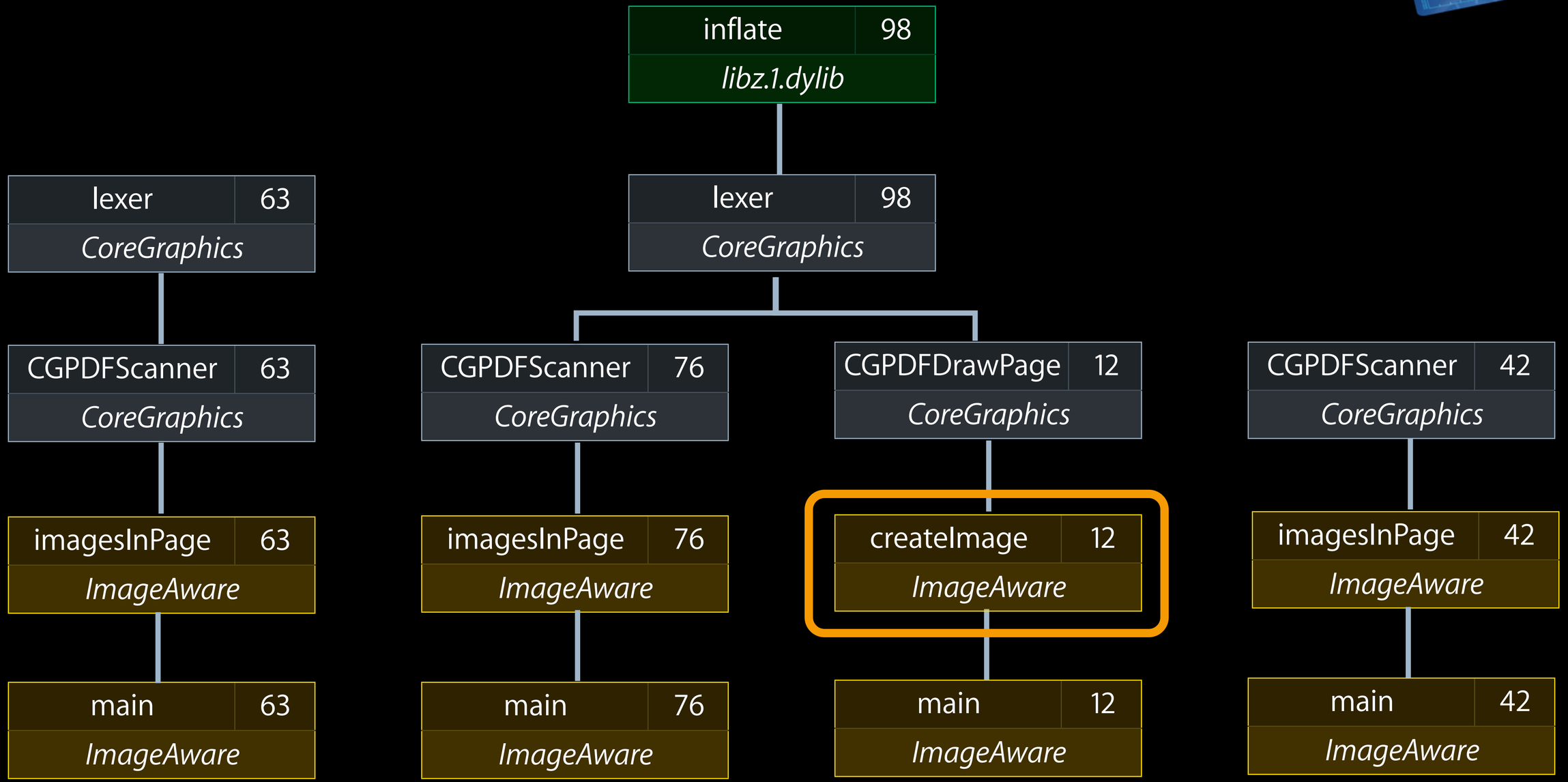
Charge Library to Caller

Data Mining



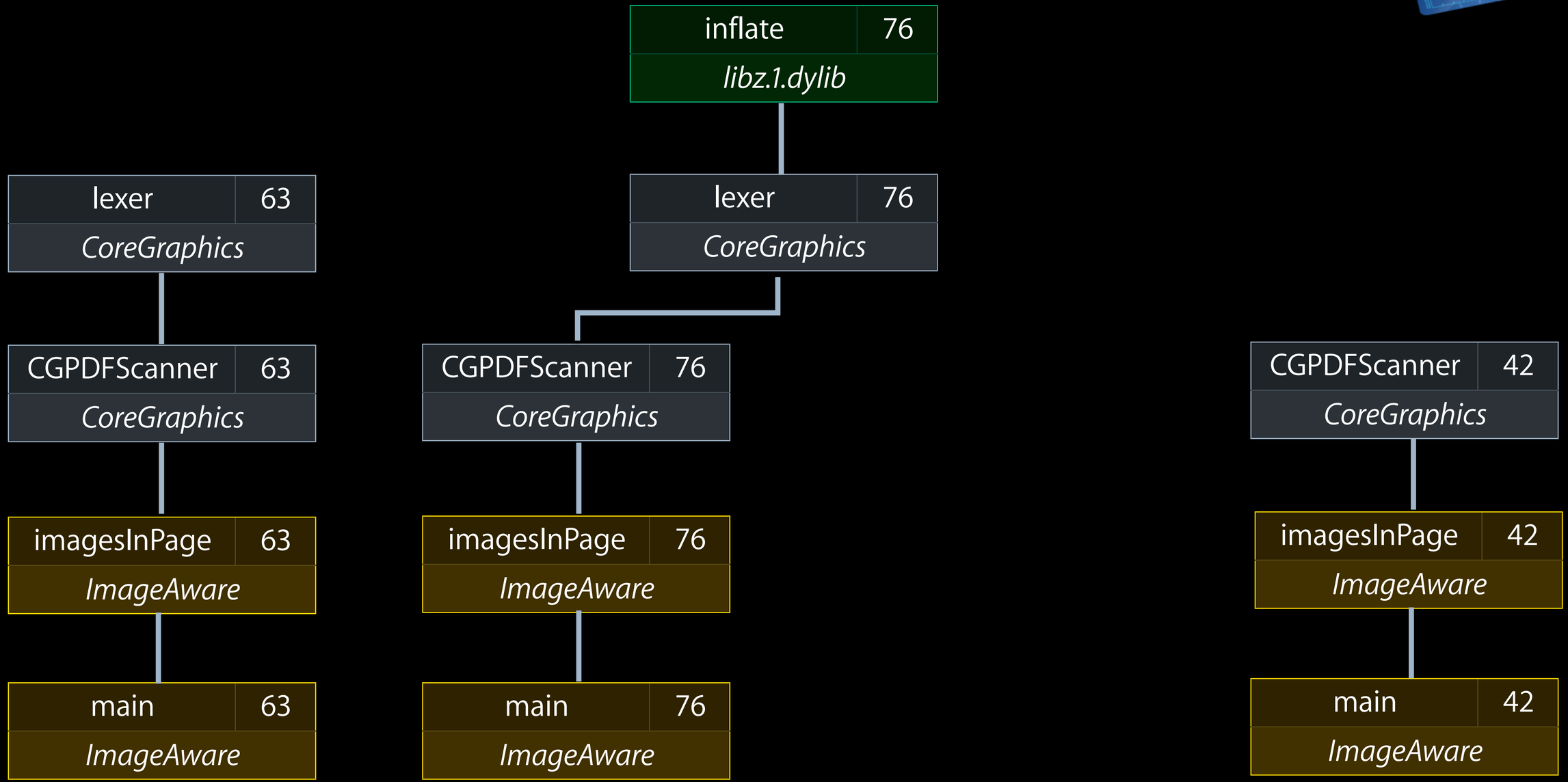
Charge Library to Caller

Data Mining



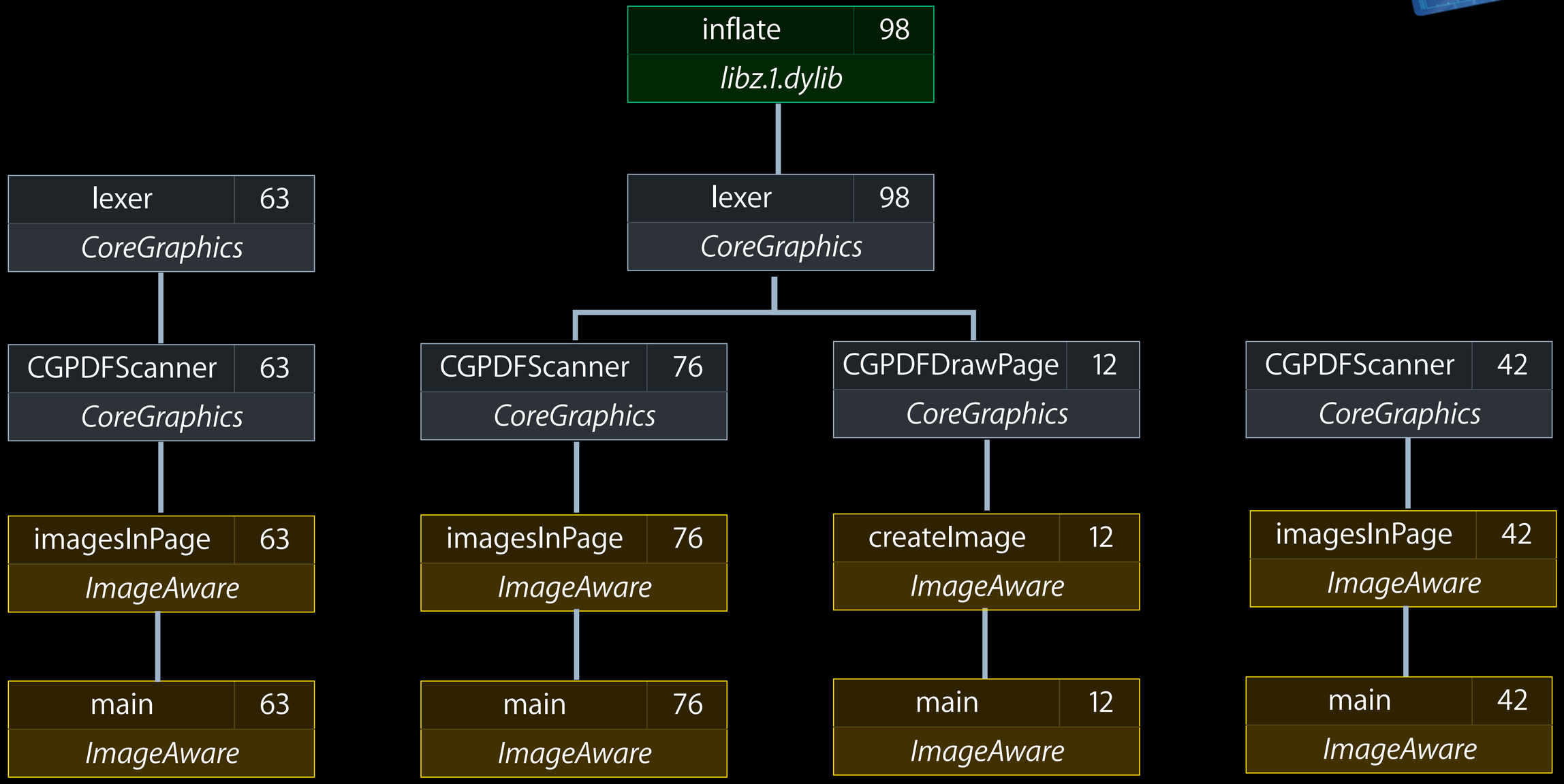
Prune Symbol and Subtree

Data Mining



Prune Symbol and Subtree

Data Mining



Flatten Library to Boundaries

Typical Views

Instruments1

Record | ImageAware.app (Default Target) | Flags | 00:00:26 Run 1 of 1 | Inspection Range | Mini | View | Library

Instruments

Time Profiler

Extended Detail

Main Thread

_class_getMethodNoSuper_nolock
libobjc.A.dylib
lookUpMethod
libobjc.A.dylib
objc_msgSend
libobjc.A.dylib
_nsnote_callback
Foundation
__CFXNotificationPost
CoreFoundation
_CFXNotificationPostNotification
CoreFoundation
-[NSNotificationCenter postNotification...
Foundation
__NSThreadPerformPerform
Foundation
__CFRunLoopDoSources0
CoreFoundation
__CFRunLoopRun
CoreFoundation
CFRunLoopRunSpecific
CoreFoundation
RunCurrentEventLoopInMode
HIToolbox
ReceiveNextEventCommon

Time Profiler

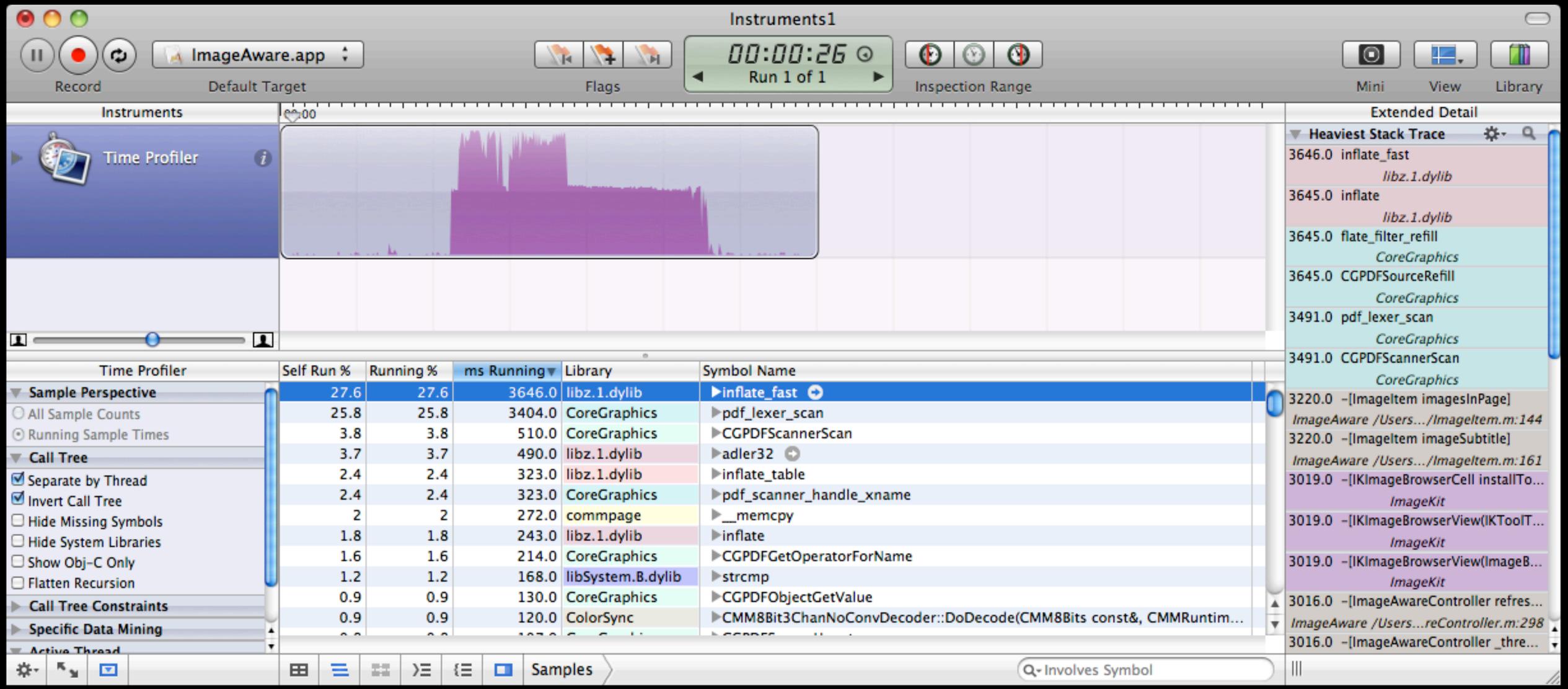
- Running Sample Times
- **Call Tree**
  - Separate by Thread
  - Invert Call Tree
  - Hide Missing Symbols
  - Hide System Libraries
  - Show Obj-C Only
  - Flatten Recursion
- **Call Tree Constraints**
- **Specific Data Mining**
- **Active Thread**

All Threads

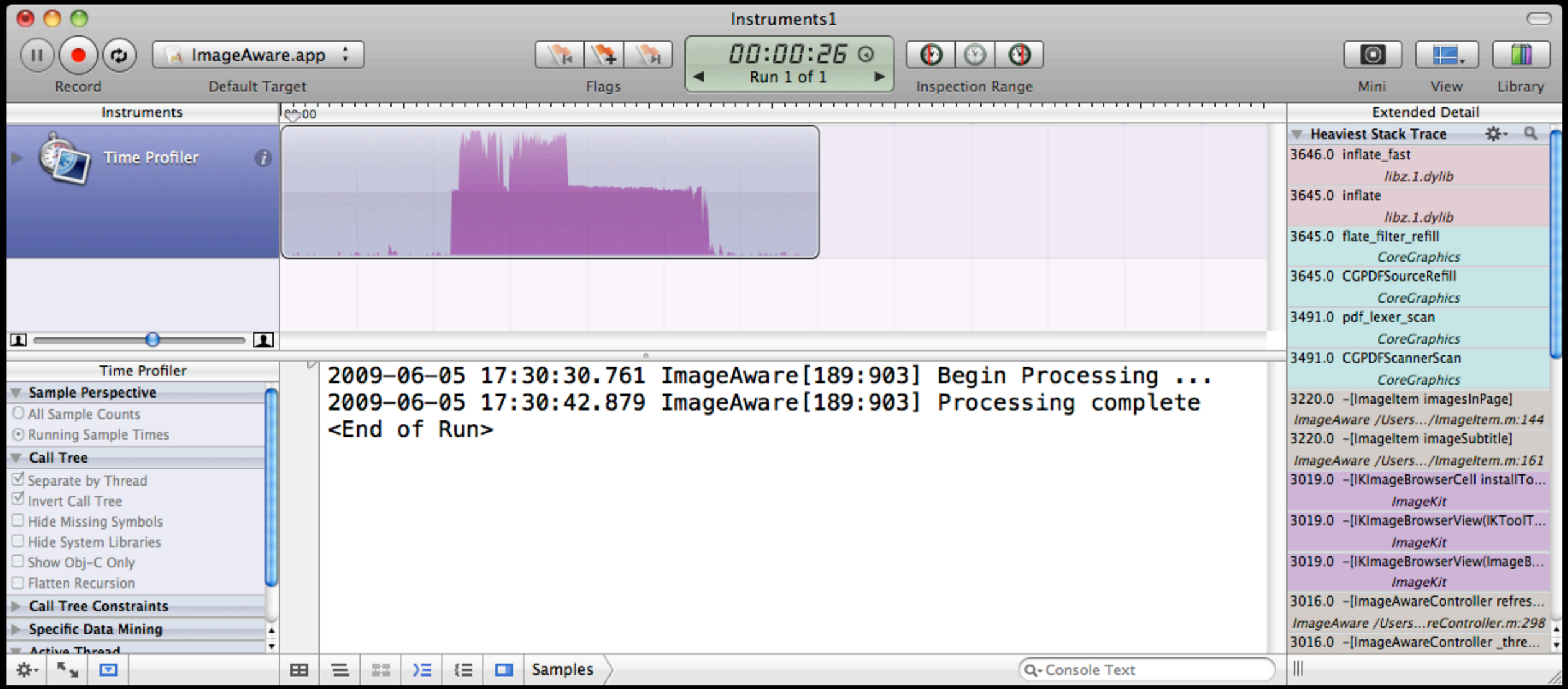
#	Timestamp	Duration	Depth	Hot Frame	Responsible Lib...	Responsible Caller
0	00:00.605	1.00 ms	20	_class_getMethodNoSuper...	Foundation	-[NSNotificationCenter pos...
1	00:00.806	1.00 ms	13	mach_msg_trap	AppKit	-[NSApplication nextEvent...
2	00:00.806	1.00 ms	9	object_cxxConstructFromC...	ImageKit	-[IKTaskManager taskLoop]
3	00:01.267	1.00 ms	12	mach_msg_trap	AppKit	-[NSApplication nextEvent...
4	00:01.607	1.00 ms	13	mach_msg_trap	AppKit	-[NSApplication nextEvent...
6	00:02.809	1.00 ms	12	mach_msg_trap	AppKit	-[NSApplication nextEvent...
6	00:02.809	1.00 ms	10	__semwait_signal	Foundation	-[NSCondition wait]
7	00:02.867	1.00 ms	11	mk_timer_arm	AppKit	-[NSApplication nextEvent...
8	00:03.410	1.00 ms	15	dyld_stub_OSSpinLockLock	AppKit	-[NSApplication nextEvent...
9	00:03.610	1.00 ms	13	CFSortIndexes	AppKit	-[NSApplication nextEvent...
10	00:03.610	1.00 ms	5	modf$fenv_access_off	ImageKit	-[IKTaskManager taskLoop]
11	00:03.711	1.00 ms	6	__error	Foundation	+[NSThread sleepUntilDate:]

Samples | Backtrace

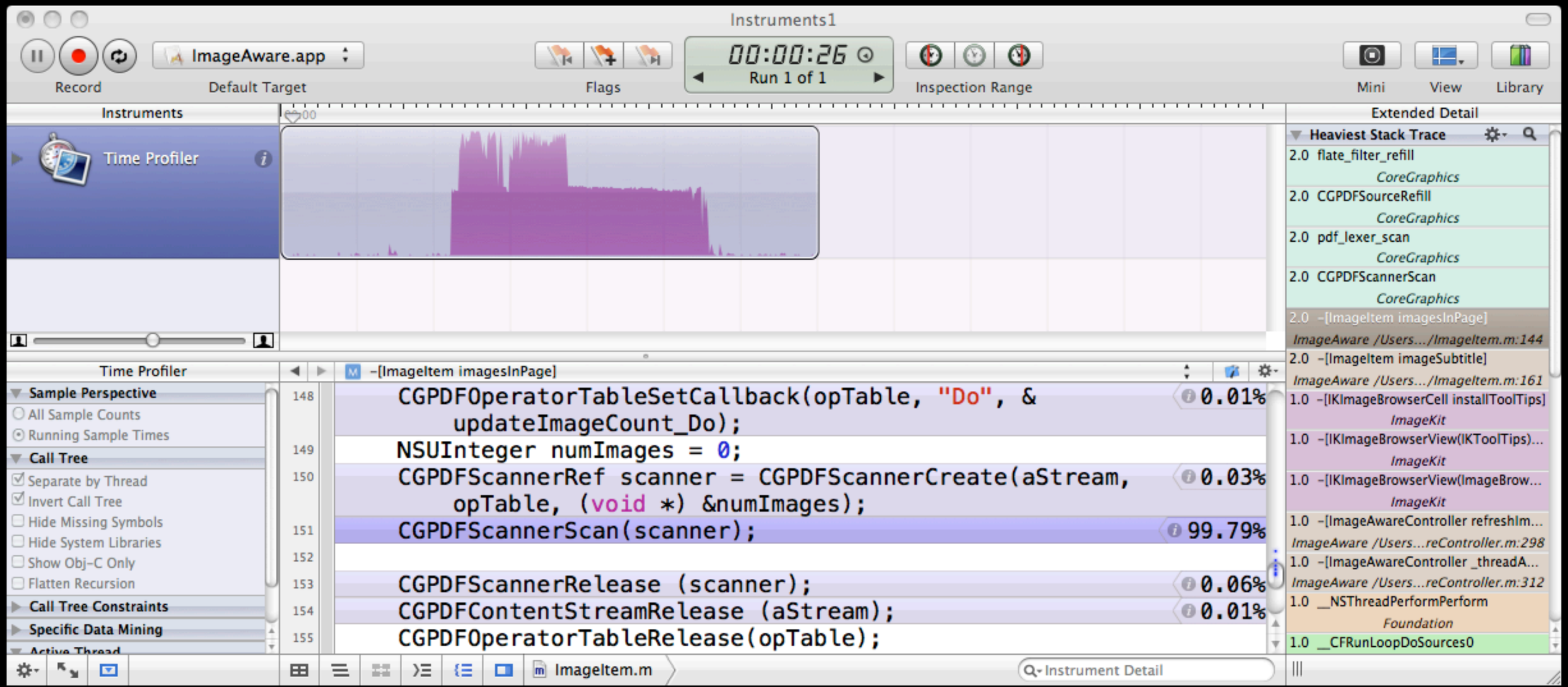
Typical Views

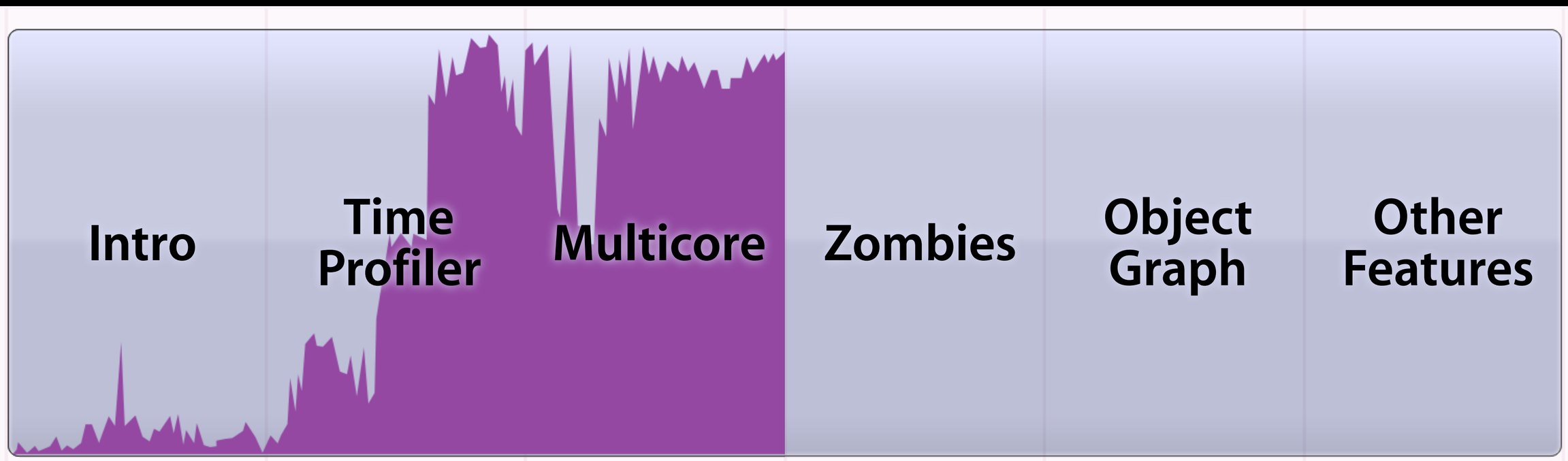


Console View



Source View


Instruments1
Record
ImageAware.app
Default Target
Flags
00:00:26
Run 1 of 1
Inspection Range
Mini
View
Library
Instruments
Time Profiler
Extended Detail
Heaviest Stack Trace
2.0 flate_filter_refill
CoreGraphics
2.0 CGPDFSourceRefill
CoreGraphics
2.0 pdf_lexer_scan
CoreGraphics
2.0 CGPDFScannerScan
CoreGraphics
2.0 -[ImageItem imagesInPage]
ImageAware /Users.../ImageItem.m:144
2.0 -[ImageItem imageSubtitle]
ImageAware /Users.../ImageItem.m:161
1.0 -[IKImageBrowserCell installToolTips]
ImageKit
1.0 -[IKImageBrowserView(IKToolTips)...
ImageKit
1.0 -[IKImageBrowserView(ImageBrow...
ImageKit
1.0 -[ImageAwareController refreshIm...
ImageAware /Users...reController.m:298
1.0 -[ImageAwareController _threadA...
ImageAware /Users...reController.m:312
1.0 __NSThreadPerformPerform
Foundation
1.0 __CFRunLoopDoSources0
Time Profiler
Sample Perspective
All Sample Counts
Running Sample Times
Call Tree
Separate by Thread
Invert Call Tree
Hide Missing Symbols
Hide System Libraries
Show Obj-C Only
Flatten Recursion
Call Tree Constraints
Specific Data Mining
-[ImageItem imagesInPage]
148 CGPDFOperatorTableSetCallback(opTable, "Do", &
updateImageCount_Do);
149 NSUInteger numImages = 0;
150 CGPDFScannerRef scanner = CGPDFScannerCreate(aStream,
opTable, (void *) &numImages);
151 CGPDFScannerScan(scanner);
152
153 CGPDFScannerRelease (scanner);
154 CGPDFContentStreamRelease (aStream);
155 CGPDFOperatorTableRelease(opTable);
0.01%
0.03%
99.79%
0.06%
0.01%
ImageItem.m
Instrument Detail


Intro
Time Profiler
Multicore
Zombies
Object Graph
Other Features

Demo

Dispatch

Multicore

Goals

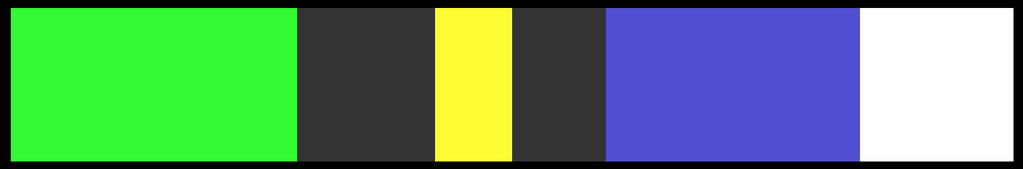
- Performance analysis
- Behavioral analysis

Threads

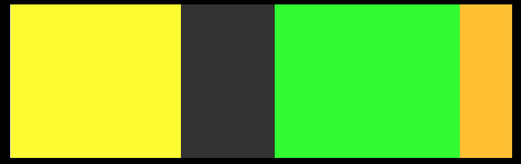
What does it show?

- Thread states

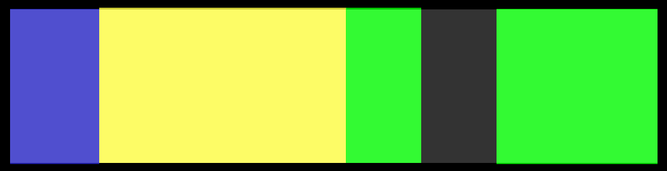
1



2



3



Waiting

Suspended

Requested to suspend

Running

On run queue

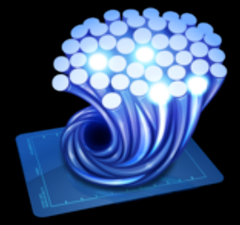
Waiting uninterruptibly

At termination

Idle

Threads

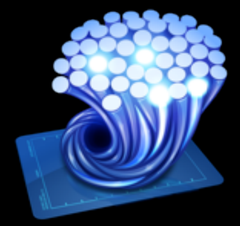
What does it show?



- Thread types
  - BSD
  - Dispatch
  - Bootstrap (main)

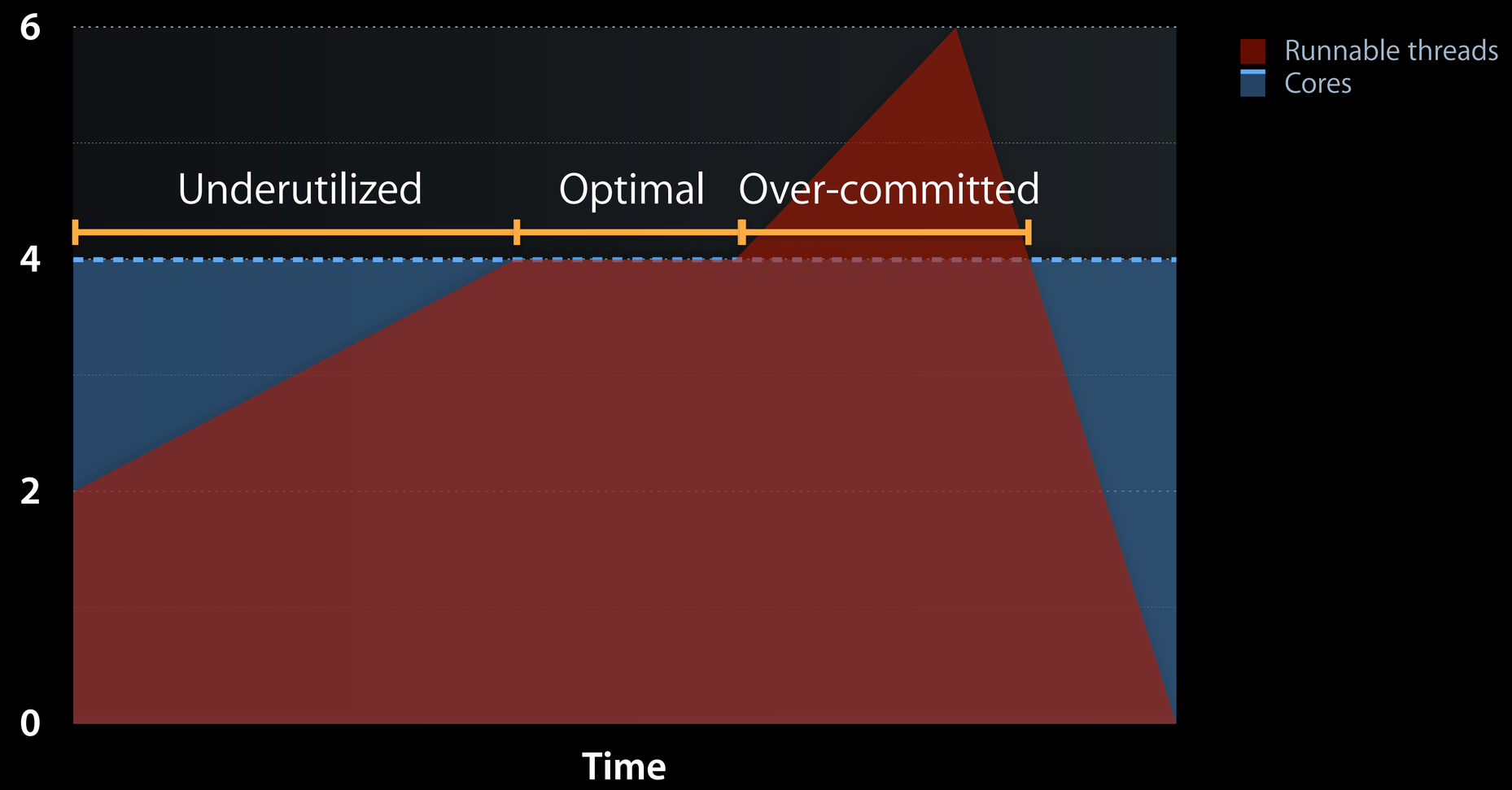
Threads

What does it show?



- Thread parent/child relationships
- Living vs. terminated
- Context switches
- On CPU time

Threads



#Threads ≈ #Cores

Dispatch

Blocks and queues

Queue	# Blocks	#Sync	Processed	Latency
queueA	5	0	0	0
queueB	3	0	0	0

Queues View

Dispatch

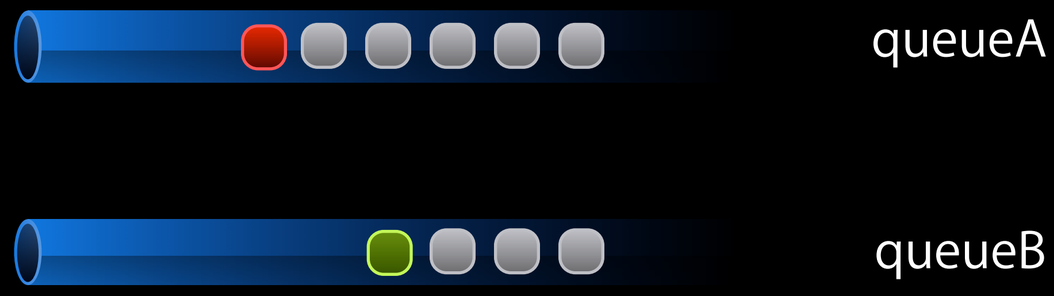
Blocks and queues

Queue	# Blocks	#Sync	Processed	Latency
queueA	6	0	0	0
queueB	3	0	0	0

Queues View

Dispatch

Blocks and queues

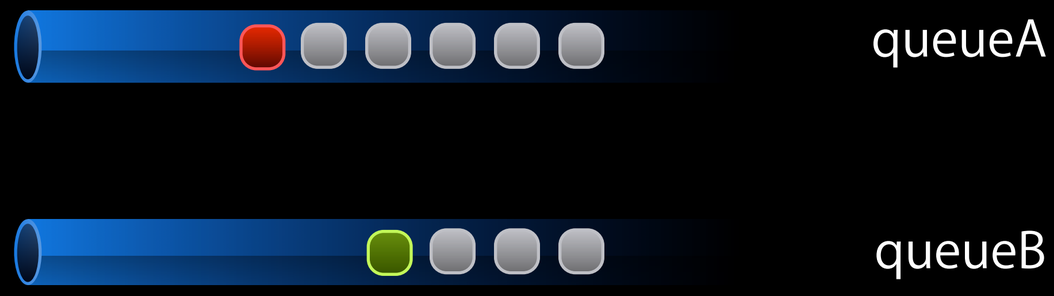


Queue	# Blocks	#Sync	Processed	Latency
queueA	6	0	0	0
queueB	4	1	0	0

Queues View

Dispatch

Blocks and queues



Queue	# Blocks	#Sync	Processed	Latency
queueA	6	0	0	0
queueB	4	1	0	0

Queues View

Dispatch

Blocks and queues

Block	# Blocks	Total Work Time	#Sync	Average Work Time	Work Time	Count	Latency
foo	5	5	0	5	1	1	43
bar	0	15	1	3	4	4	56
baz		9		2		4	

Queues View

Blocks View

Dispatch

Nested Blocks

```
dispatch_block_t bar = ^{
        baz();
};
```

```
dispatch_block_t baz = ^{
    //do something
};
```

Block	Total Work Time	Average Work Time	Count
foo	5	5	1
bar	15	3	4
baz	9	2	4

Blocks View

Dispatch

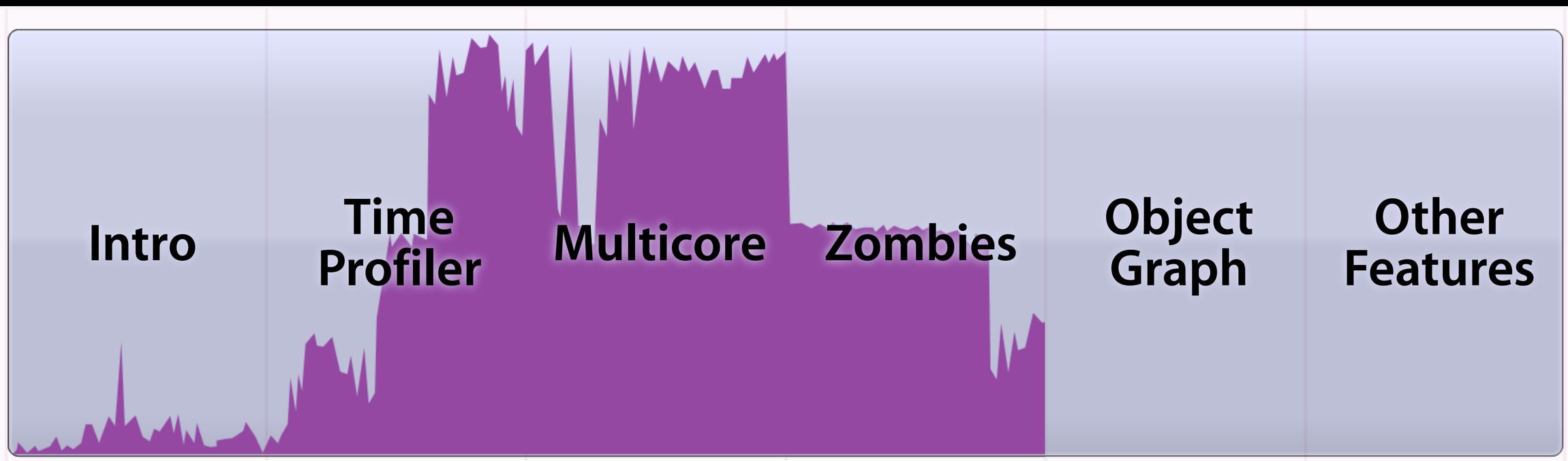
Blocks performance

- Optimize Blocks
  - Longest duration
  - Most executed

Dispatch

Queue behavior

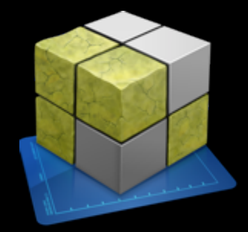
- Analyze queues
  - Block execution order
  - Latency
  - Synchronous API

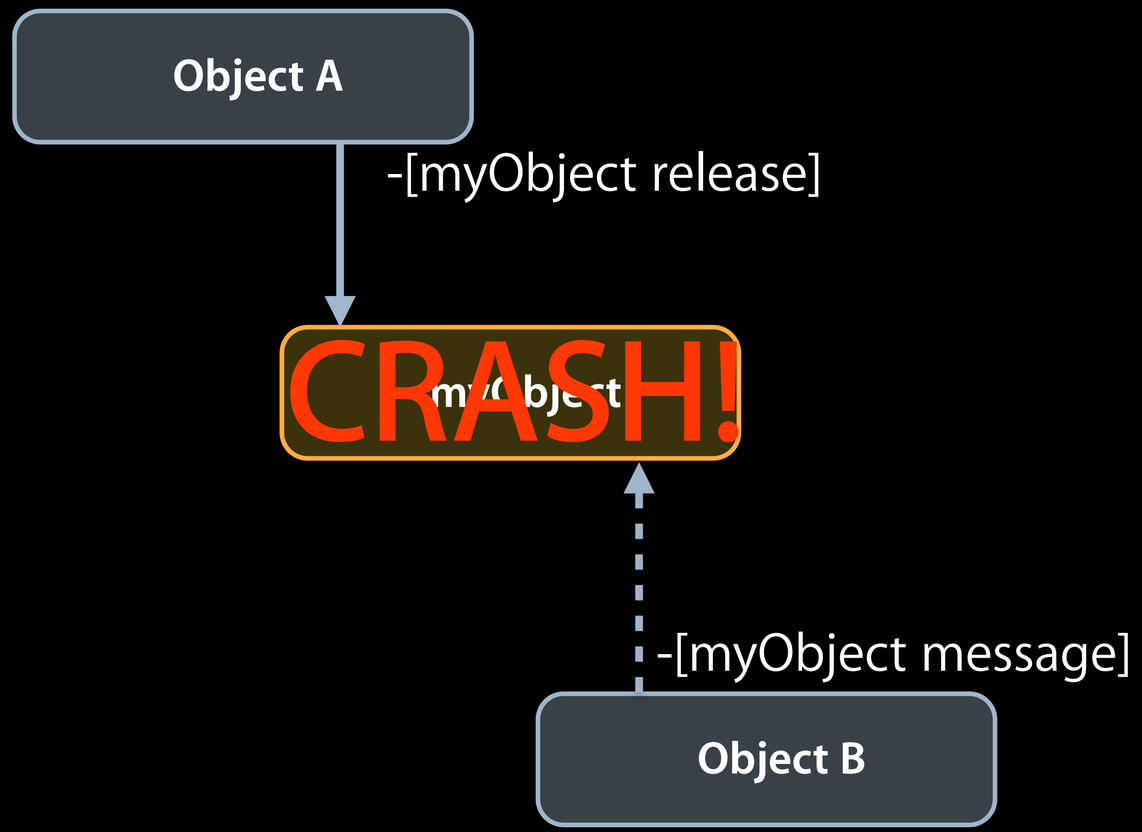


Daniel Delwood
Performance Tools Engineer

Over-Released Objects

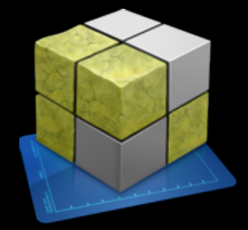
What are Zombies?

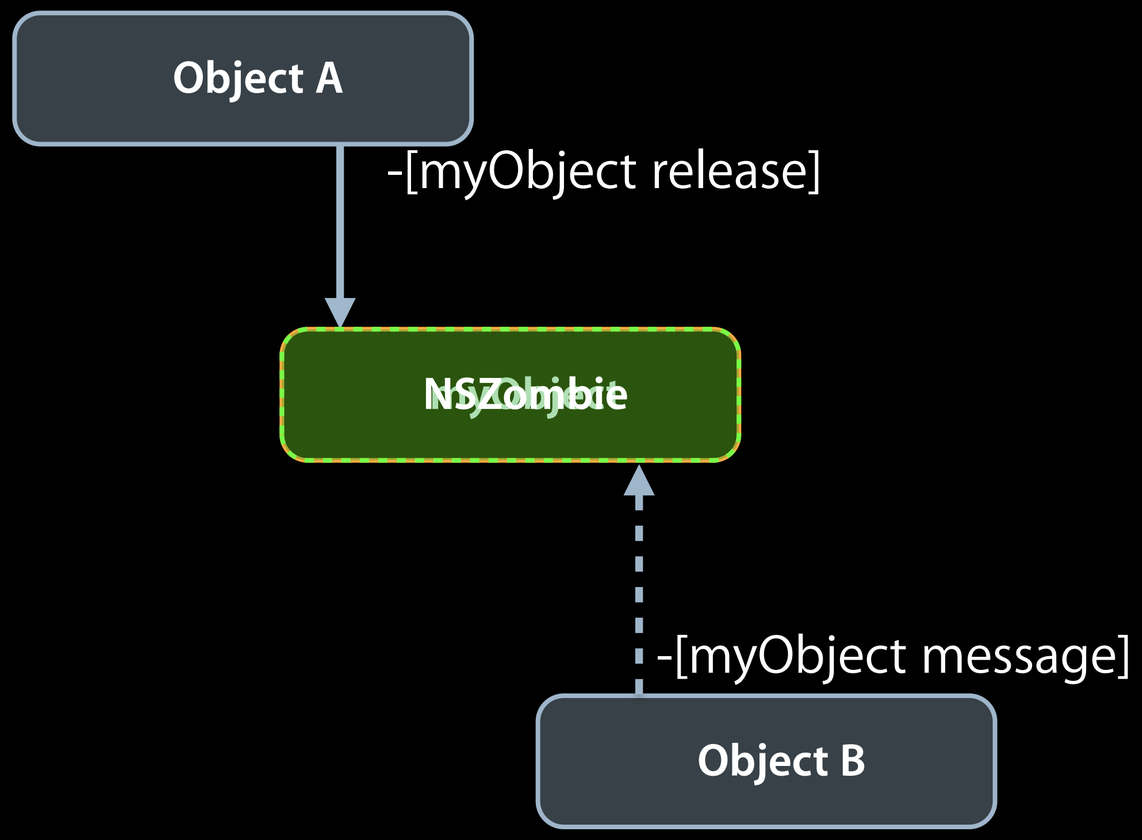




Over-Released Objects

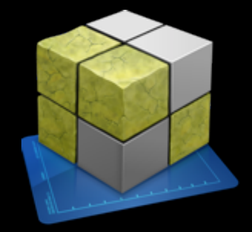
What are Zombies?



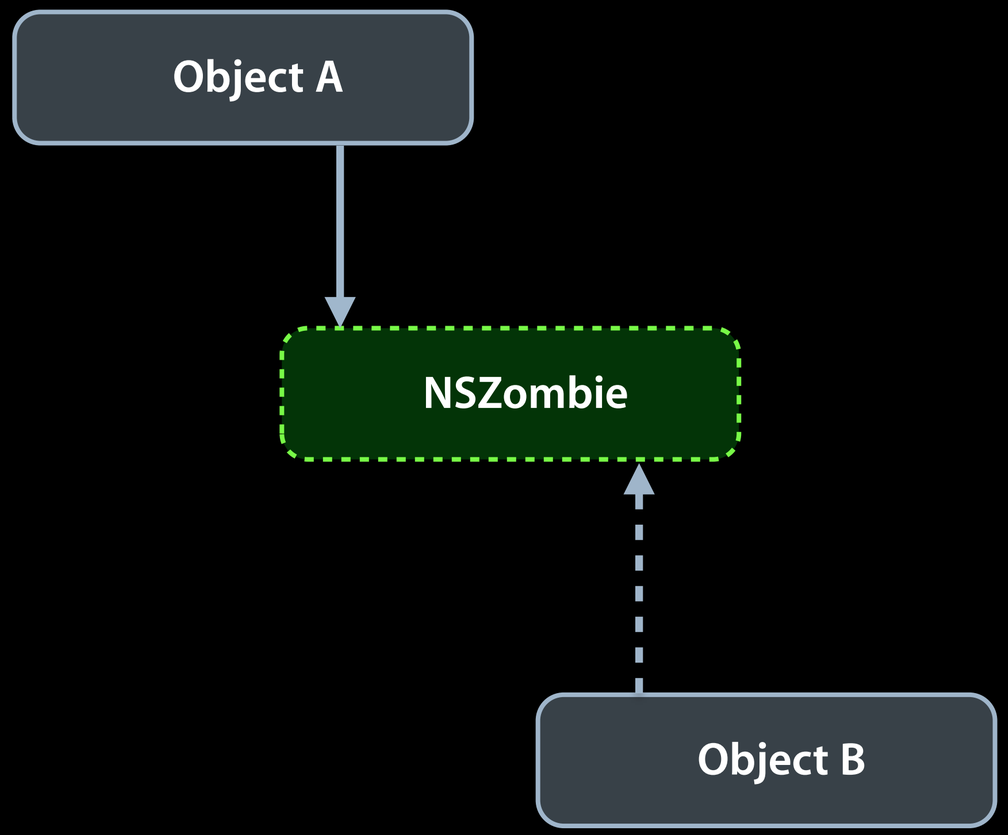


Over-Released Objects

Zombies template

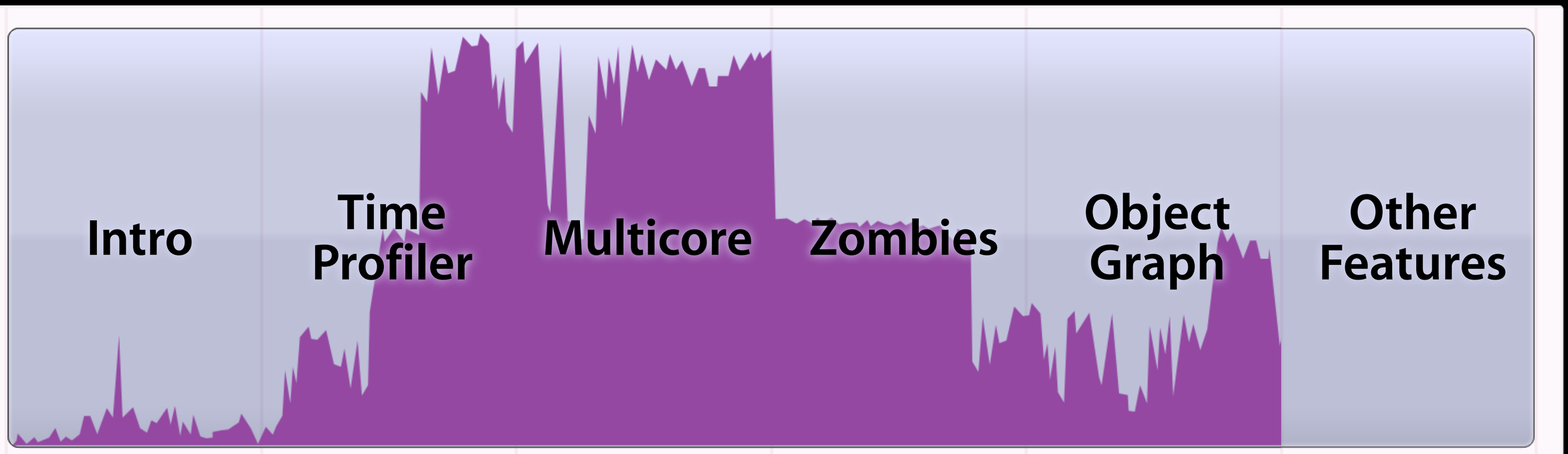


- Detects Zombie message
- Zombie retain/release/autorelease history



Demo

Zombies


Intro
Time Profiler
Multicore
Zombies
Object Graph
Other Features

Garbage Collection

Memory management simplified

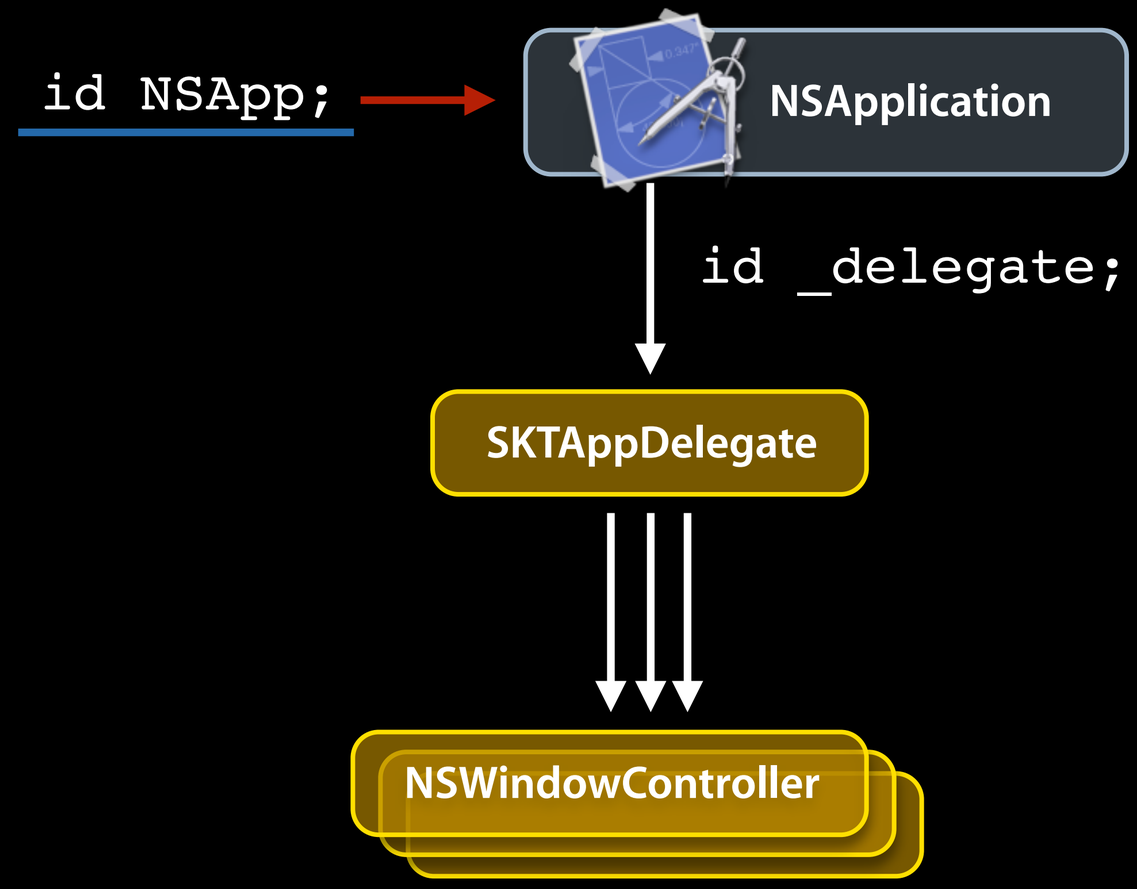
- Retain/release unnecessary
- Life cycles managed by references

Unnecessary references → unnecessary memory

Object Graph

Manage references

- Understand “roots”
- Causes of a root
- Path from object to root

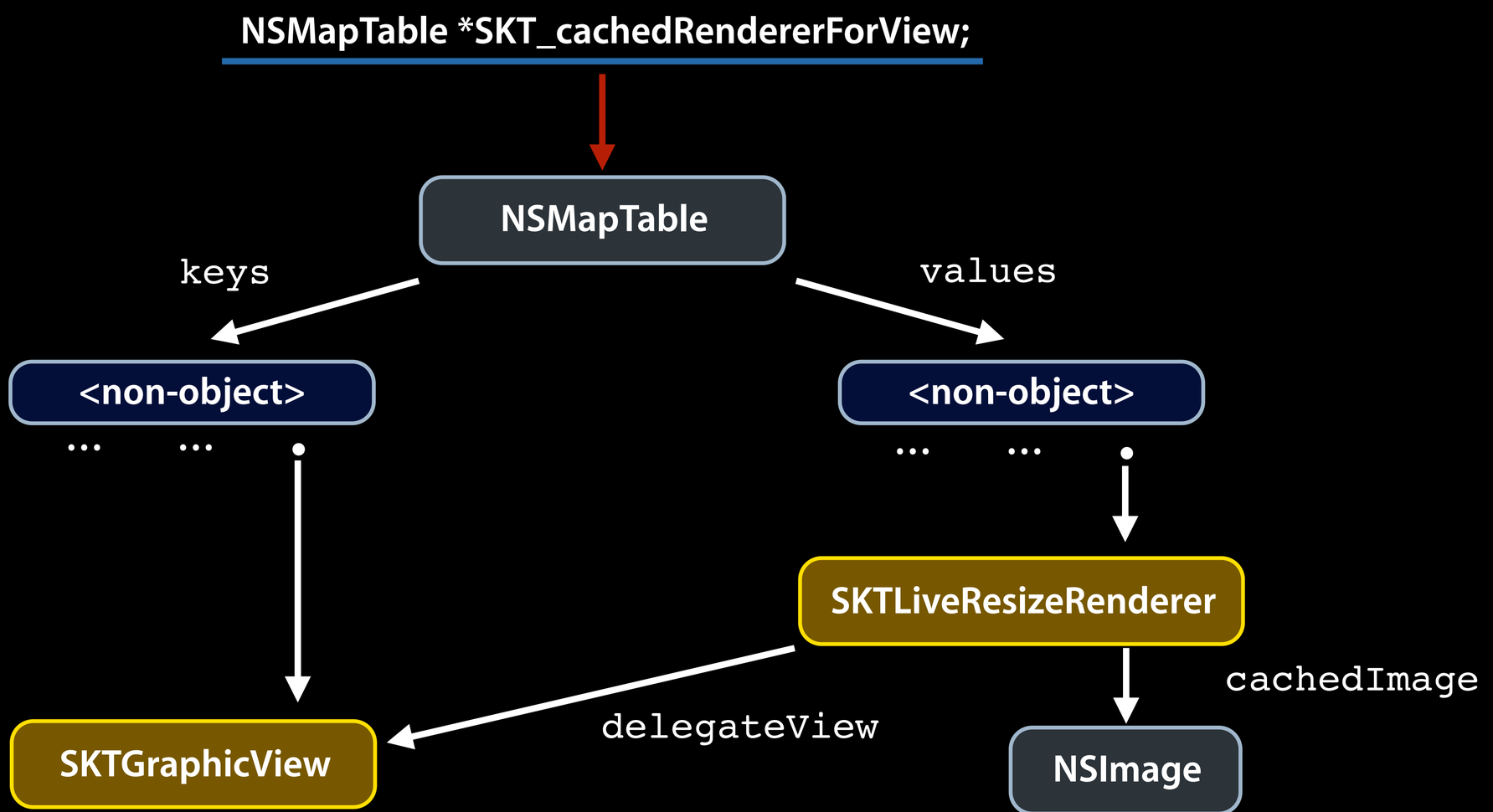


Demo

Object Graph

Object Graph

SKTGraphicView never dies



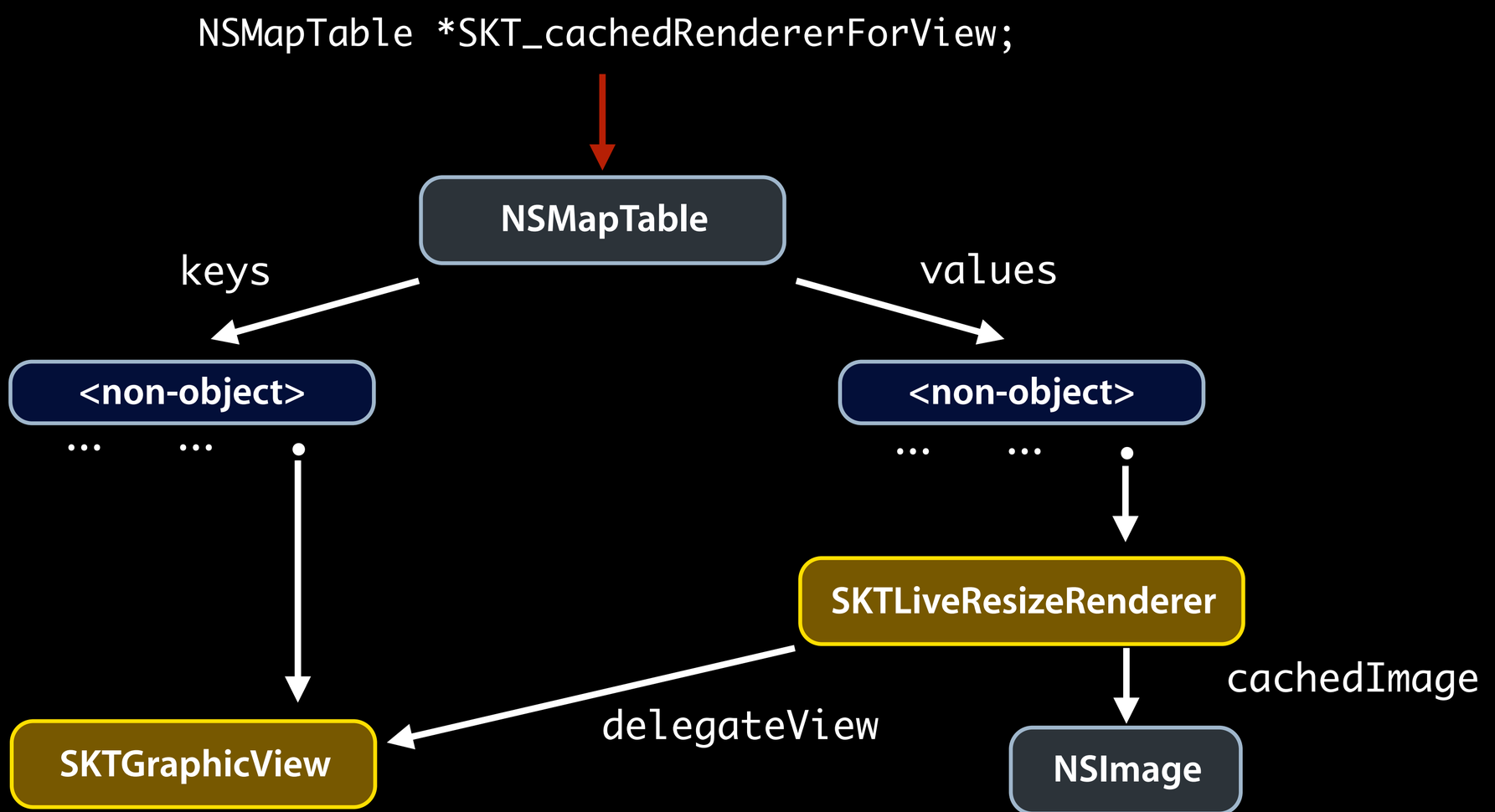
Object Graph

SKTGraphicView never dies

```
[[NSMapTable alloc] initWithKeyOptions:NSPointerFunctionsZeroingWeakMemory
        valueOptions:NSPointerFunctionsObjectPersonality capacity:0ul];
```

Object Graph

SKTGraphicView never dies



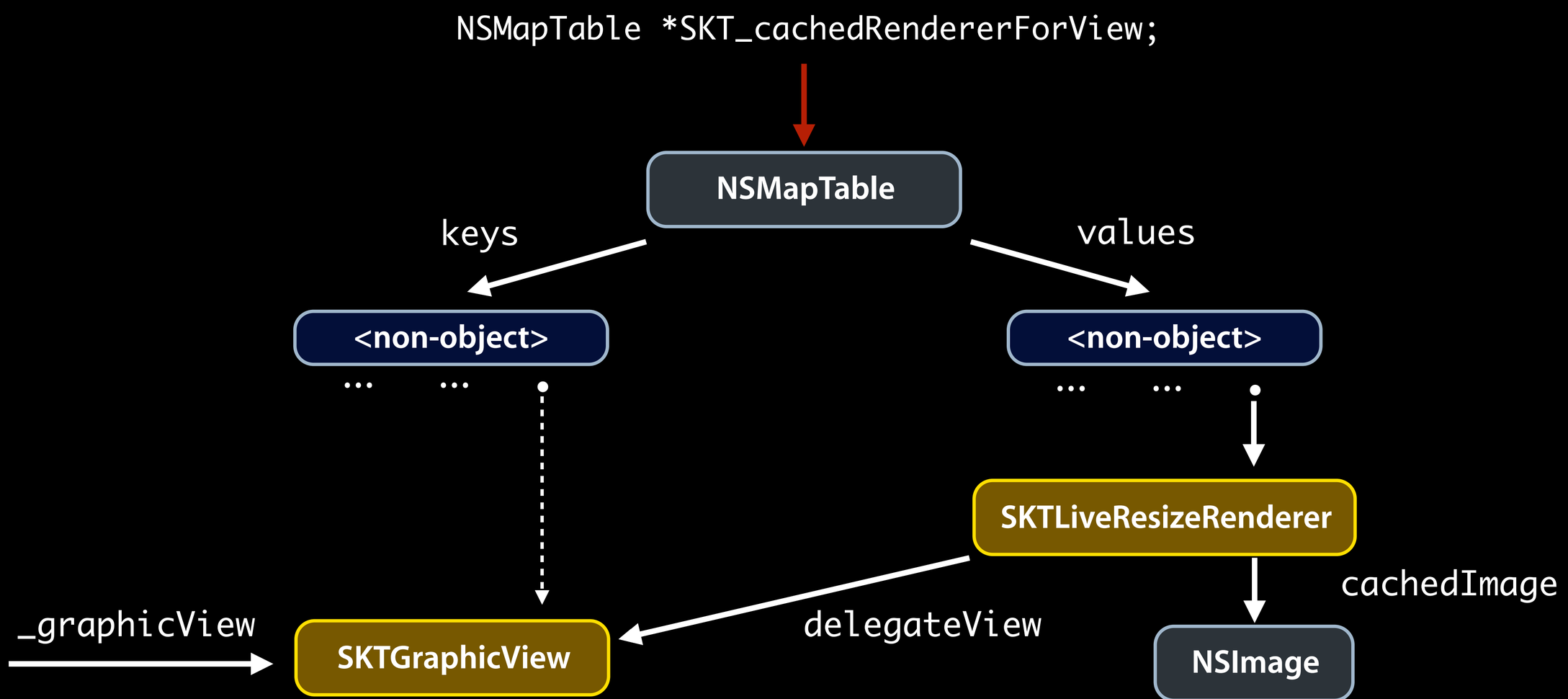
Object Graph

SKTGraphicView never dies

```
@interface SKTLiveResizeRenderer : NSObject {
    ...
    NSView __weak *delegateView;
}
```

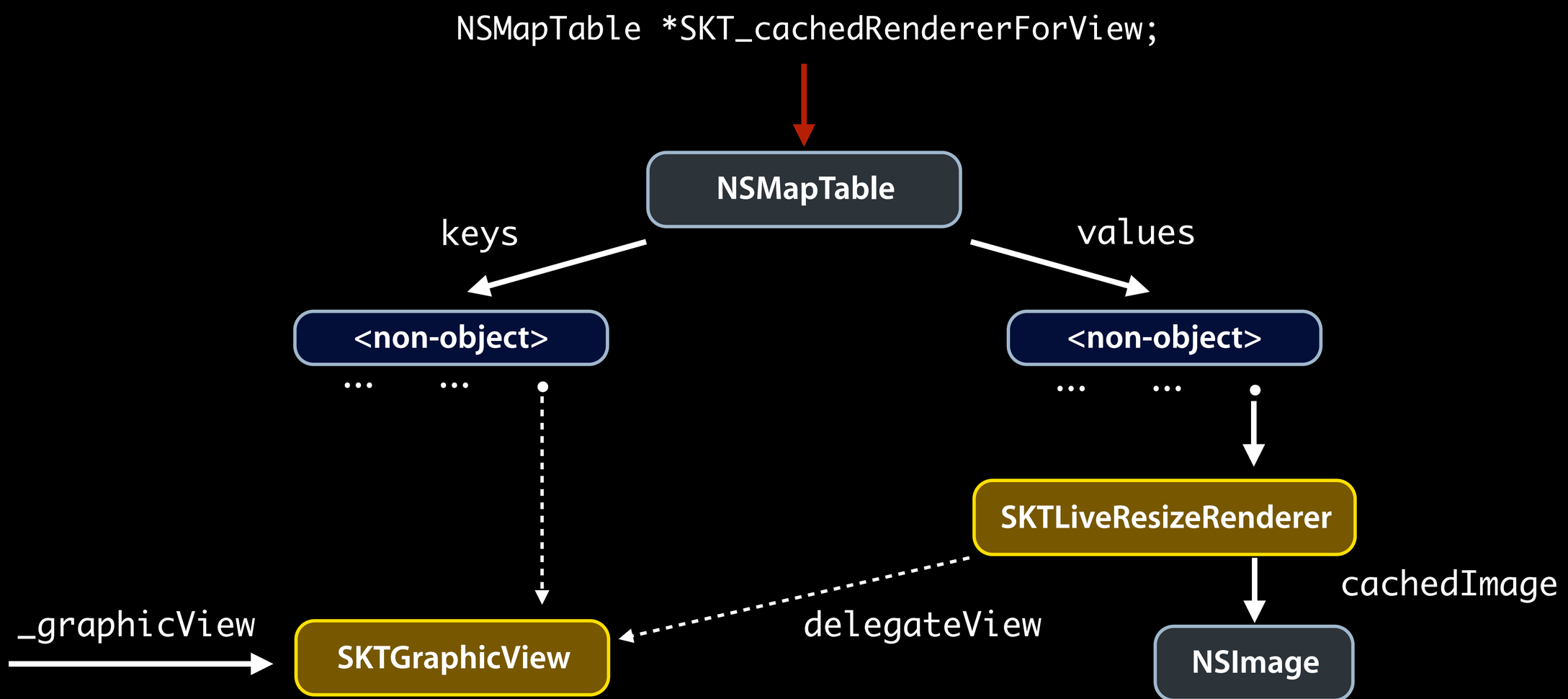
Object Graph

SKTGraphicView never dies



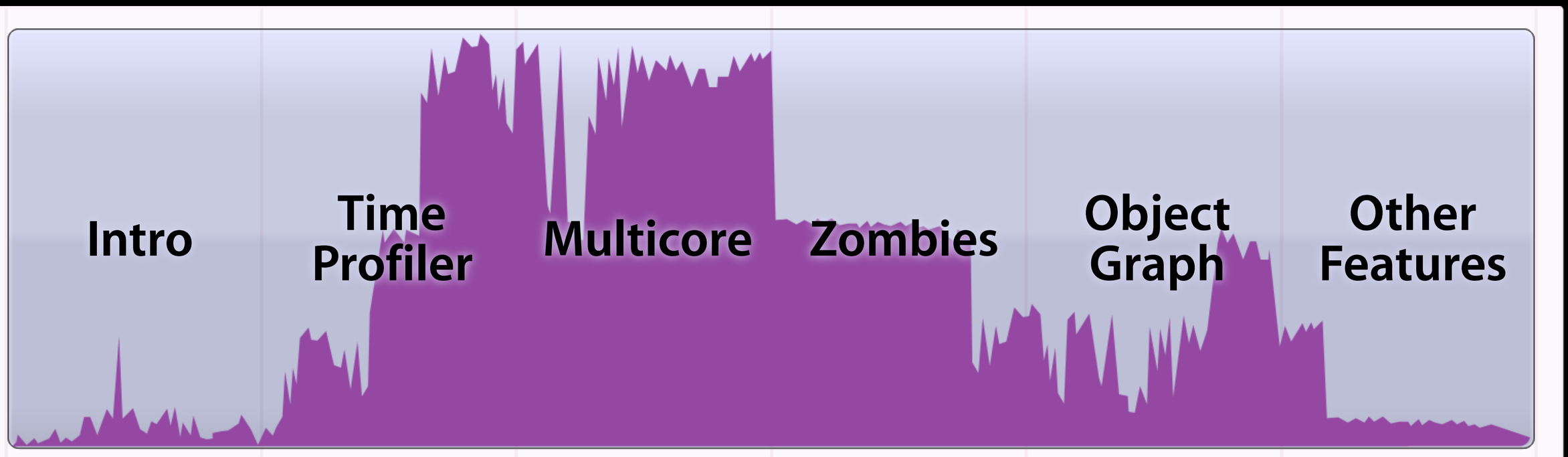
Object Graph

SKTGraphicView never dies



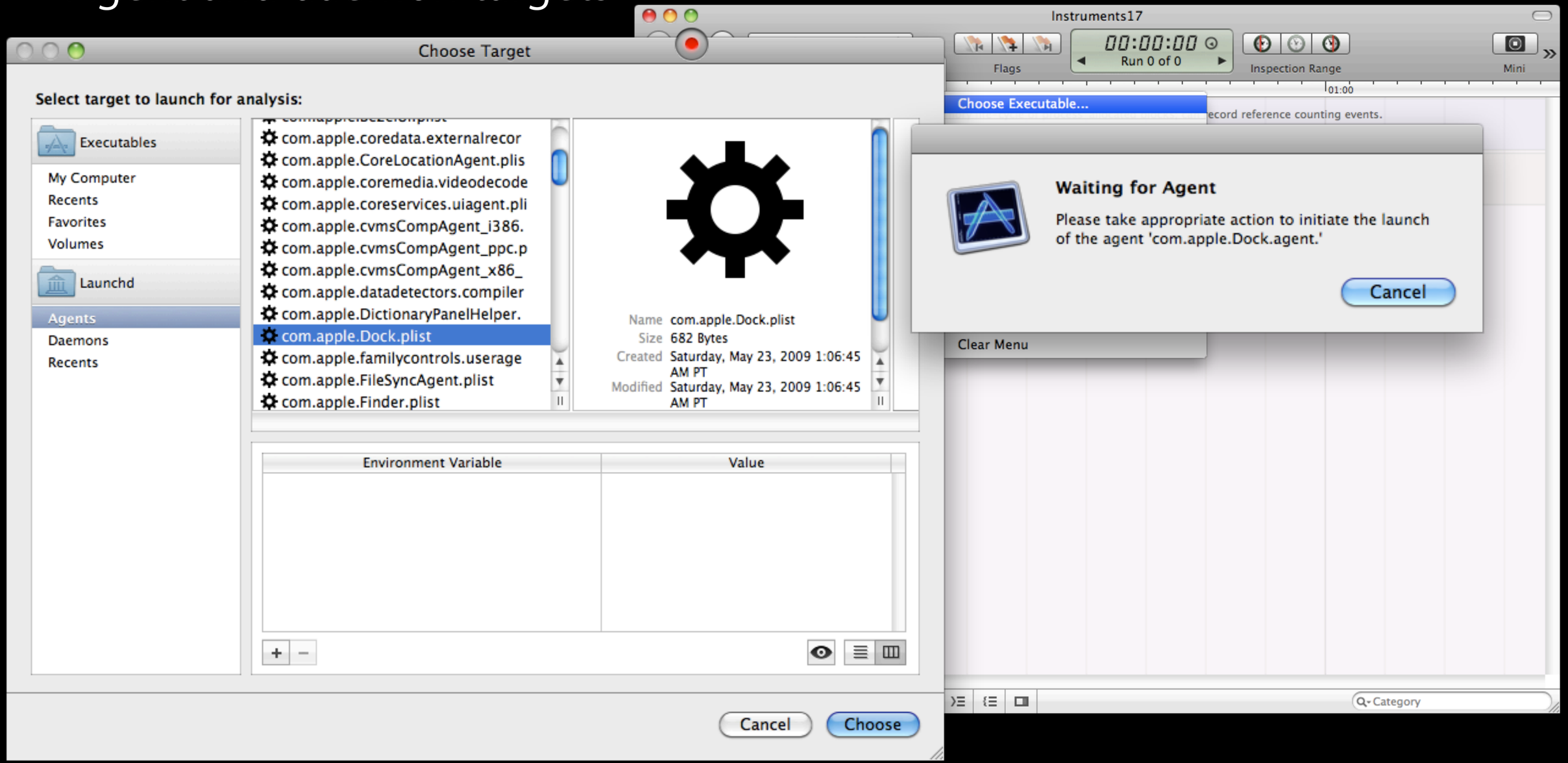
Object Graph

- Understand GC object relationships
- Identify unnecessary dependencies
- Eliminate over-rootedness


Intro
Time Profiler
Multicore
Zombies
Object Graph
Other Features

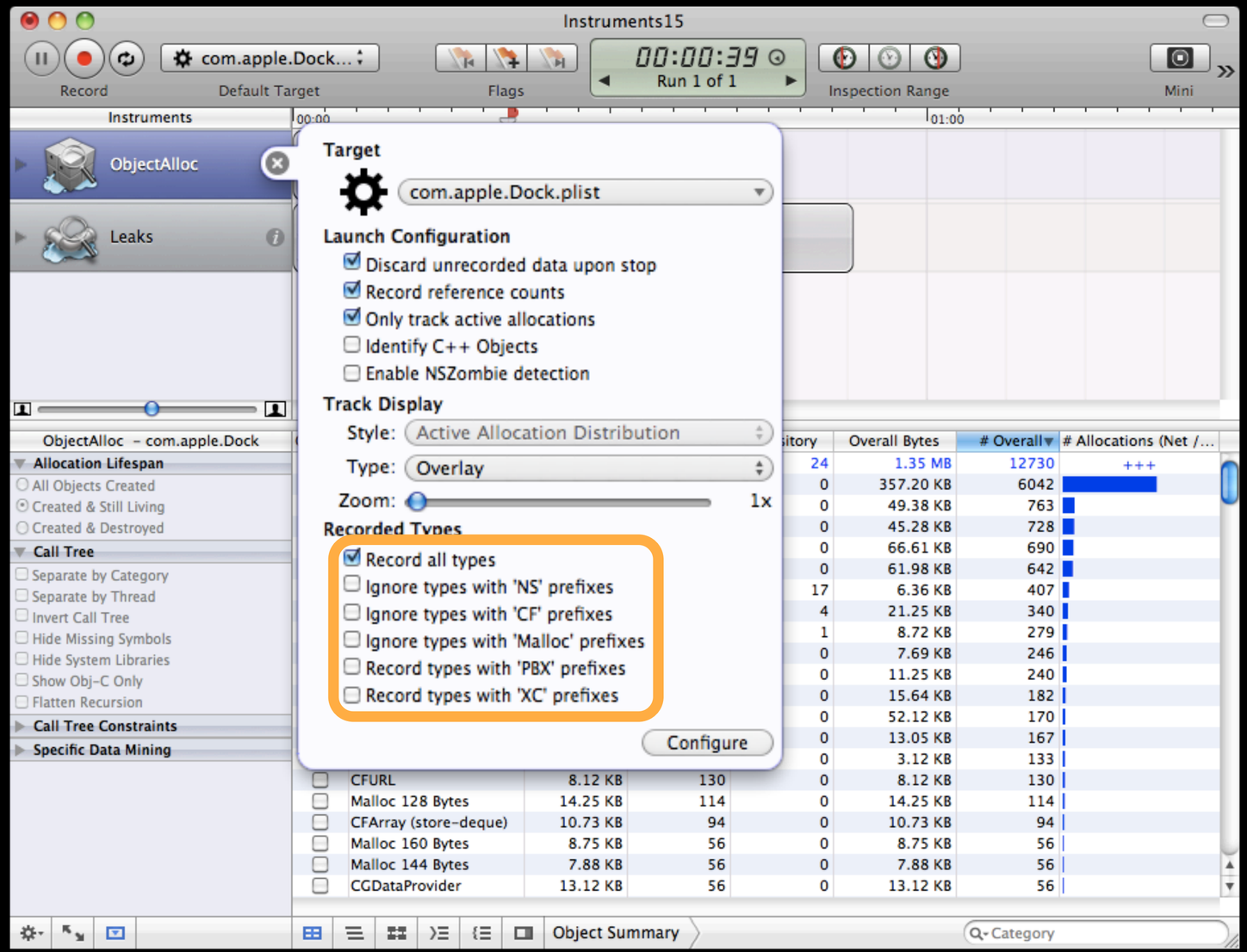
Other Features

- Agent and daemon targets



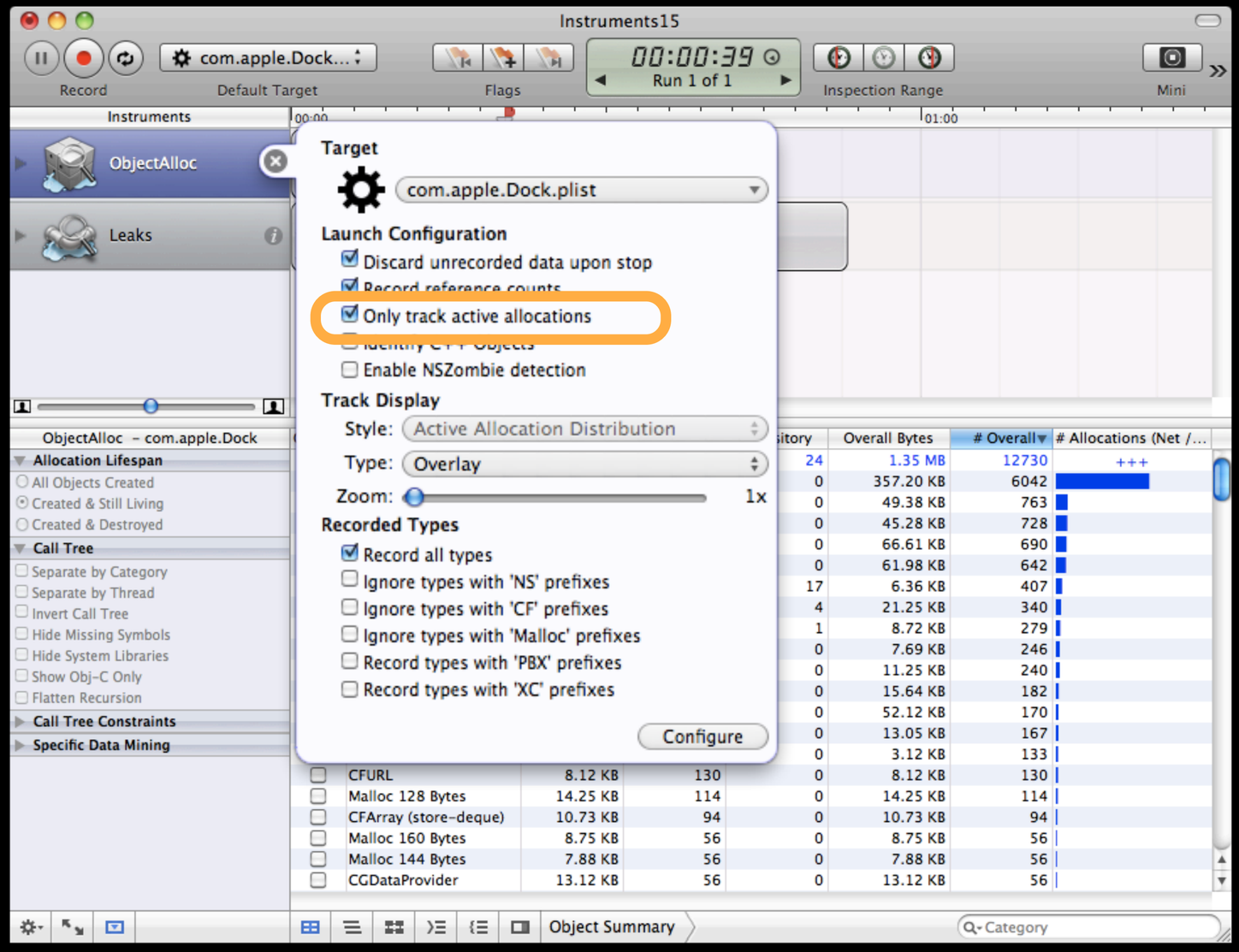
Other Features

- Agent and daemon targets
- ObjectAlloc filtering



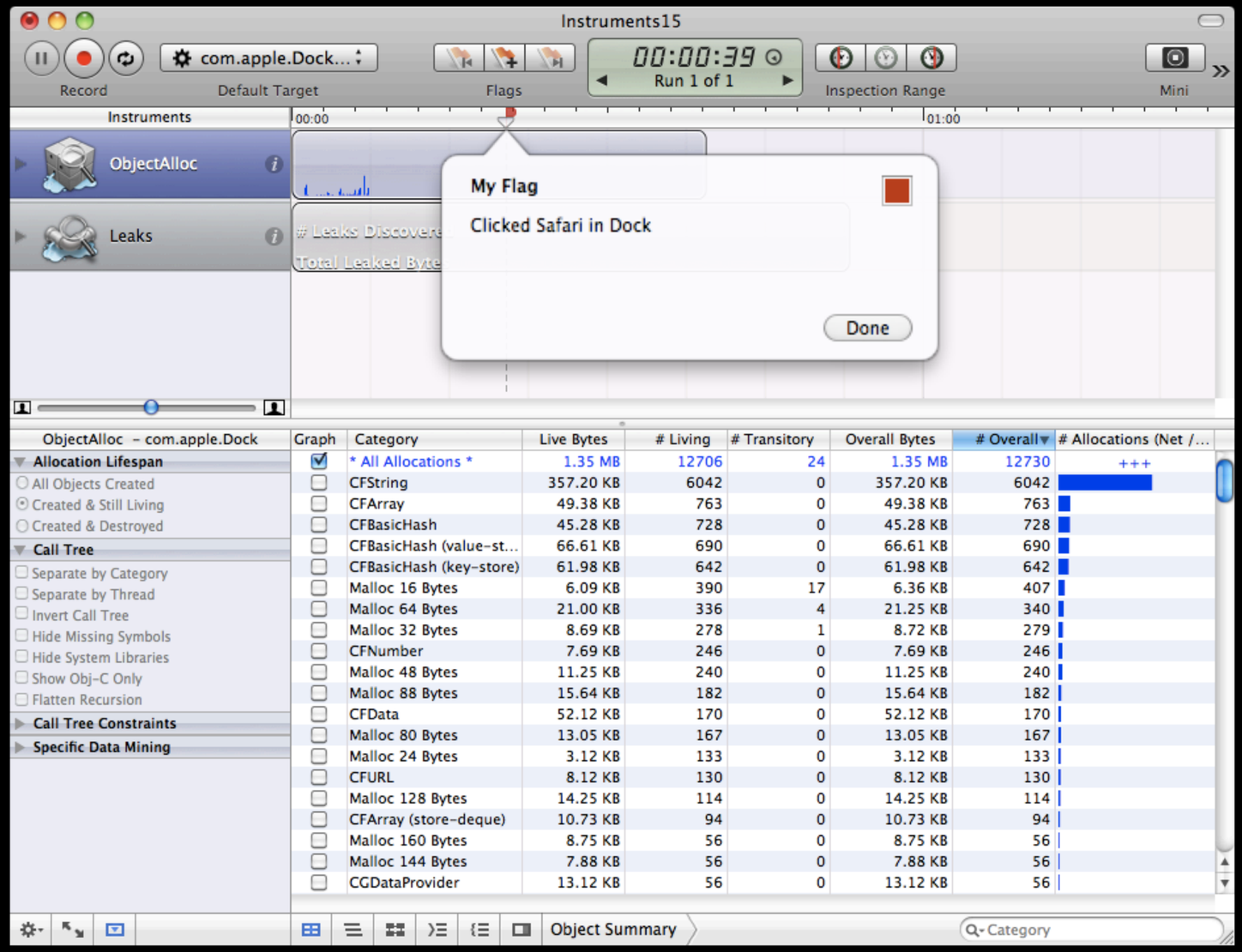
Other Features

- Agent and daemon targets
- ObjectAlloc filtering
- Leaks throw away life cycle complete objects



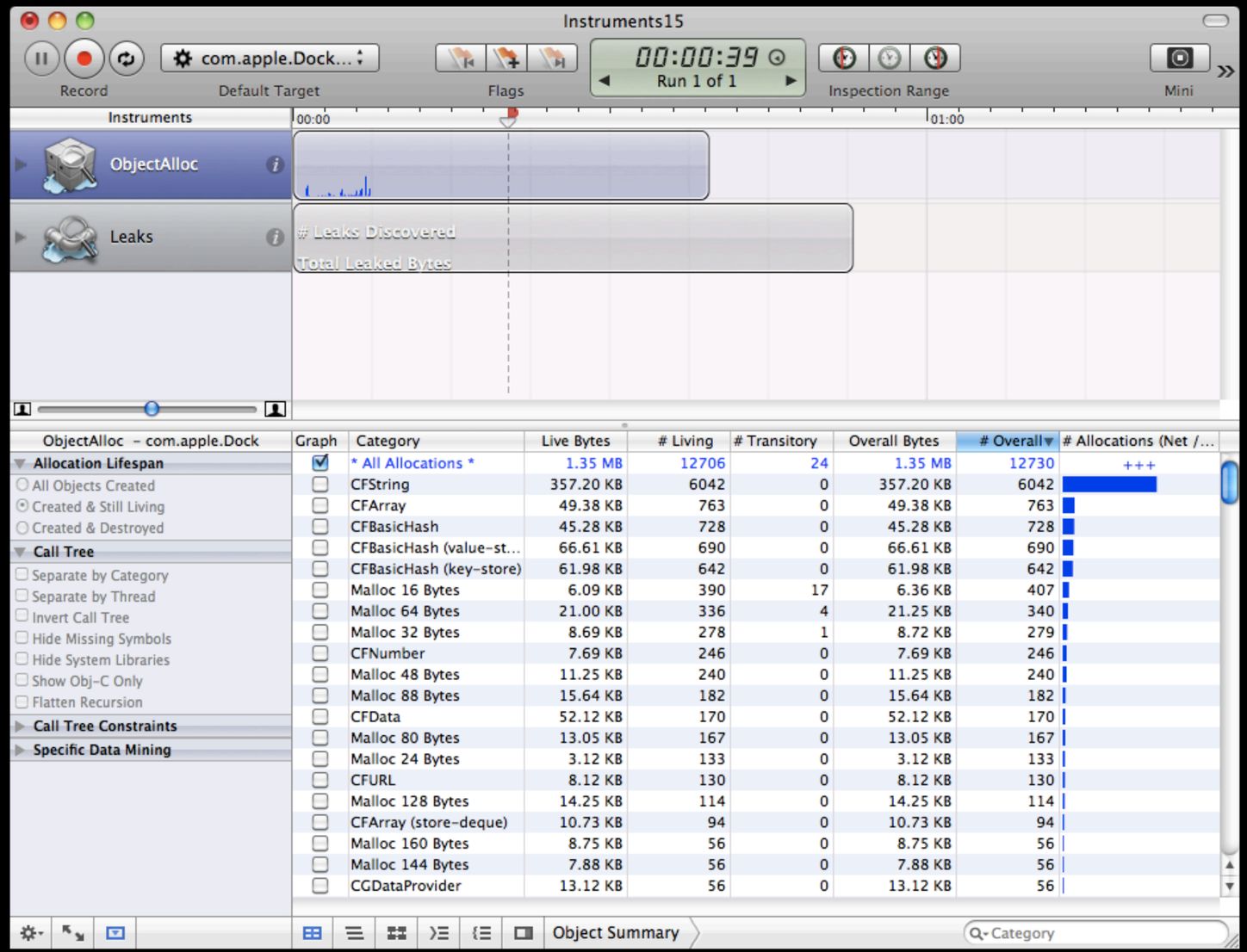
Other Features

- Agent and daemon targets
- ObjectAlloc filtering
- Leaks throw away life cycle complete objects
- Flags

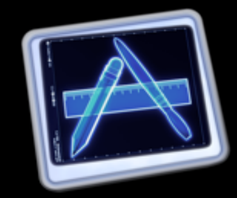


Other Features

- Agent and daemon targets
- ObjectAlloc filtering
- Leaks throw away life cycle complete objects
- Flags
- Faster symbolication
- Symbols for lazily loaded libraries



Conclusion



Instruments: Powerful analysis tool

- New instrumentation
- Enhanced data collection
- Better performance

More Information

Michael Jurewitz
Developer Tools Evangelist
jurewitz@apple.com

Instruments Documentation
Instruments User Guide (Xcode documentation)

Concurrency Programming Guide
Xcode documentation

Related Sessions

Advanced Debugging and Performance Analysis	Nob Hill Thursday 9:00AM
Understanding Grand Central Dispatch in Depth	Presidio Thursday 9:00AM
Understanding Memory Management on iPhone	Mission Thursday 3:30PM
iPhone Performance Optimization with Instruments	Marina Thursday 5:00PM
Performance Tuning with Shark on Mac and iPhone	North Beach Friday 9:00AM

Related Labs

Grand Central Dispatch Lab	Snow Leopard Lab Wednesday 3:30PM
Advanced Debugging and Performance Analysis Lab	OS X Lab B Thursday 9:00AM
iPhone Performance Optimization with Instruments Lab	iPhone Lab B Friday 2:00PM

Q&A

WWDC Session Survey

You can rate your sessions on the WWDC Attendee Site.

Survey forms are located on each session description page at:

developer.apple.com/wwdc/attendee

WWDC09